An Architecture, Design and Implementation of a Communications Server to Access Disparate Databases

by

FRANCIS C.K. GAN

Submitted to
the Department of Mechanical Engineering
in Partial Fulfillment of
the Requirements of The Degree of

Bachelor of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1989

Signature of Author _____
Department of Mechanical Engineering
May 8, 1989

Certified by _____
Stuart E. Madnick
Professor, Management Sciences
Thesis Supervisor

Accepted by _____
Peter Griffith
Professor, Department Committee
Mechanical Engineering Department

An Architecture, Design and Implementation of a Communications Server to Access Disparate Databases

by

FRANCIS C.K. GAN

Abstract:

Composite Informations Systems are aimed at increasing connectivity among disparate databases. The Composite Information Systems/Tool Kit (CIS/TK) research project at the Massachusetts Institute of Technology Sloan School addresses four related aspects of connectivity, specifically strategic, organizational, physical and logical connectivity. This thesis presents an improved solution to the physical connectivity problem.

The Local Query Processor (LQP) module of CIS/TK provides the interface to each information resource, local or foreign. The module in the existing CIS/TK system of concern is the Communications Server (CS), a part of the LQP. In implementing the new CS, the internal structure of the LQP has been modified slightly with the CS being modularized and removed from the structure of the LQP. LQPs for non-UNIX and non-SQL DBMSs are now much easier to implement with the new "respond to prompt" CS.

The new CS is tested for its modularity by making it totally compatible with existing LQPs and also for a new LQP implemented in this thesis, the I. P. Sharp/Disclosure database.

Thesis Supervisor: Stuart E. Madnick

Title: Professor of Management Science

Table of Contents

Table of Figures

Acknowledgements/Dedications

This thesis caps off my four years at MIT and more than anything else, reflects that most of what I have learnt at MIT was outside class. Professor Madnick's continuous encouragement played a big role in my learning as much as I did in the process of completing my thesis. Not lacking at all was the help and technical assistance I received from the rest of the CIS/TK group, with special thanks to Mia who helped me start on the right foot in the beginning and has put up with my never-ending "first cut" implementations.

I thank my mother for her love, never ending guidance and encouragement, without which I would not be where I am today. Not to forget Dad, from whom I have inherited some good traits which have very important in my life. My deepest appreciation goes to all my relatives in Malaysia especially my grandparents, Peter and Annie Cho, for all the love and support over the last 22 years and for their pride in me that has helped me through the last four years. I am deeply honored by all of you travelling halfway round the earth to attend my graduation and wedding.

For the love of my life and fiancée, Emilia, I give you all my love for just being there with your love, company, and encouragement in the last three years. I'm looking forward to our wedding on June 10th to top off the best week in my life so far.

1. Introduction

One important category of strategic applications involves inter-corporate linkage or intra-corporate integration. Such applications require access to and integration of disparate databases. A Composite Information System (CIS) is an information system that is aimed at increasing the connectivity among disparate databases. An implementation of a CIS is the Composite Information System /Tool Kit (CIS/TK), a research project being done at the Massachusetts Institute of Technology Sloan School.

1.1 The Composite Information Systems/Tool Kit

CIS/TK's approach to increasing connectivity among information systems addresses four related aspects of connectivity, specifically strategic, organizational, physical and logical connectivity. CIS/TK is able to retrieve and consolidate data from various different types of databases running on different platforms and located worldwide. Using artificial intelligence, object-oriented programming, networking and DBMS technologies, CIS/TK can process the query and retrieve the necessary data from various sources and consolidate the data into the required form or order.

1.2 The Current CIS/TK Architecture

The CIS/TK query processor architecture is shown in Figure 1. The architecture consists of an Application Query Processor (AQP), a Global Query Processor (GQP), and a Local Query Processor (LQP) to interface with each information resource, local or foreign.

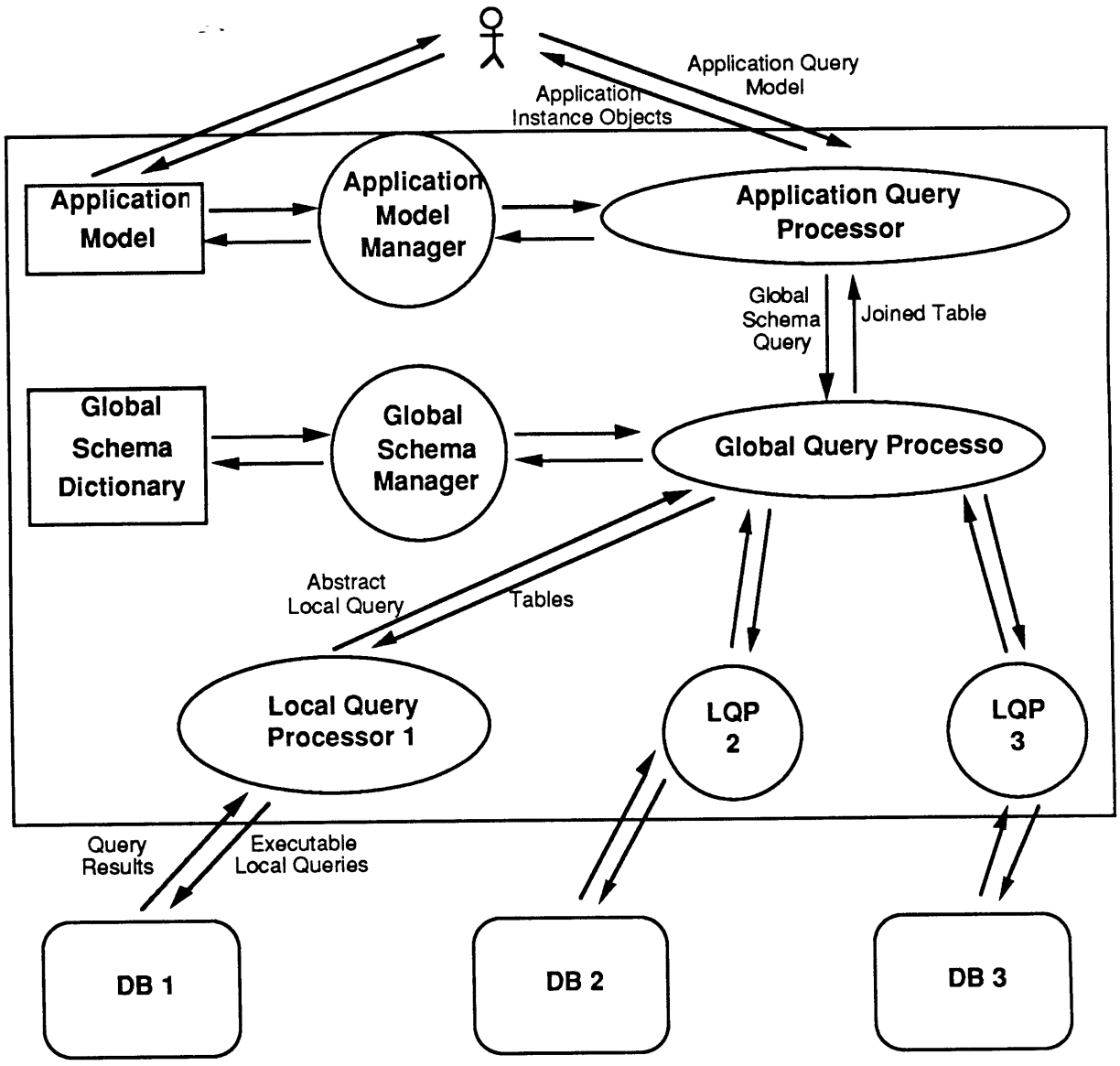


Figure 1: CIS/TK Architecture

The module of concern in CIS/TK to this paper is the Local Query Processor (LQP), that provides the interface to currently available, independent Data Base Management Systems (DBMS's). The LQP establishes the physical connection between the host and the appropriate remote machines where information is stored, transforms the abstract local query into the appropriate executable query commands for the remote system, sends the executable query commands to the actual processor (eg. the DBMS's), receives the results, and transforms the results

to the standard GQP format[1]. Each LQP currently consists of all the communication and translation routines necessary to act as a virtual driver for the desired database(s). At the time of the writing of this thesis, there exists LQP's that support *donner* (an IBM PC/RT), *mit2a, mit2c* and *mit2e* (all AT&T 3B2's). Currently under development are LQP's for I.P. Sharp's Disclosure database (the second part of this thesis), Reuter's Dataline database[2] and a database in SQL/DS on an IBM 4341[3].

1.3 CIS Issues

The first order issues in connectivity problems[4] can be thought of as annoying and inconvenient problems caused by differences in physical connections and syntax of commands. This issue has been addressed partially already[5] but this paper will try to carry it further. In moving towards the goal of providing a dynamic platform for supporting timely knowledge delivery and information intensive applications, this thesis addresses the issue of the time it takes to develop an LQP for a new DBMS.

1.4 Goals of this Thesis

This thesis provides the first step of a solution by addressing the shortcoming/limitation of the LQP in the Communications Server (CS) section. The communications server provides the actual mechanism for connecting to and invoking the remote database. It also provides a mechanism for placing the

---

[1] definition from Connectivity Among Information Systems, Y. Richard Wang and Stuart E. Madnick in CACM, Computing Practice

[2] Howard Gerber, Thesis, May 1989

[3] Gautam, Thesis, May 1989

[4] as defined by Y. Richard Wang and Stuart E. Madnick in Connectivity Among Information Systems submitted to CACM, Computing Practice

[5] Alec Champlain, Bachelor's thesis May, 1988

DBMS results into a file on the local machine. The central issue which this thesis addresses is the robustness and efficiency of the communications server.

1.5 Overview of This Thesis

Chapter 2 will discuss the limitations of the current LQP design and limitations of the CS, and propose changes in both. The design and implementation of a new communications server will be covered in detail in Chapter 3 while Chapter 4 goes on to show the use of the communications server through the implementation of a non-SQL, non-UNIX, non-networked[6] database system, I.P Sharp. Appendix A contains the newly developed code for both the communications server and the I.P. Sharp LQP. Appendix B contains the modified versions of existing code to make them either more efficient or more compatible with the new components. Appendix C contains instructions on how to use the new communications server. Appendix D shows a sample session of the I.P. Sharp LQP running using the new communications server.

[6] not networked in the sense that it cannot be accessed via telnet but instead by phone dialup.

2. Current State of the LQP

2.1 Issues in the existing CIS/TK LQP structure

As mentioned in Chapter 1, first order connectivity problems are the problems caused by differences in physical connections and syntax of commands. Such problems exist as long as information resources are dispersed across geographic location (intra- or inter-organizational) and multiple vendor machines are used. For example, at the MIT Sloan School, the recruiting database is stored on an IBM PC/RT whereas the alumni database is stored on an AT&T 3B2. Until now, the CIS/TK system has only been accessing SQL databases on UNIX based systems via Telnet. Prior to this, it has been relatively difficult and time consuming to implement an LQP for other kinds of systems. This difficulty is partly due to the fact that the current LQP structure was designed with the above mentioned UNIX/SQL systems in mind. The main problem lay in design of the communications server and its location within the structure of the LQP. Restructuring the LQP design will make it easier for new LQP's to be created for any system. The new design should enable each LQP to share the same Communications Server (CS), the software module that performs the actual communication or physical connection. This new CS should be utilized by the three new LQP's being developed concurrently with this CS.

The old structure of the LQP was not very modular at all. Although the functions of some of the components of an LQP would be the same across machines, only feeble attempts were made to modularize the similar functions and make them independent of the LQP or type of host machine. For example, the SQL translator and the LQP drivers (named connect.lsp in Appendix B). As shown in Figure 2 below, each LQP had its own parts. If any of the parts could be

shared, they would essentially have to be copied and modified for each different LQP. If the foreign hosts were similar or alike, that would not pose much of a problem but in general, database hosts are not alike and writing any section that did the same thing would require more tailoring and additional work.

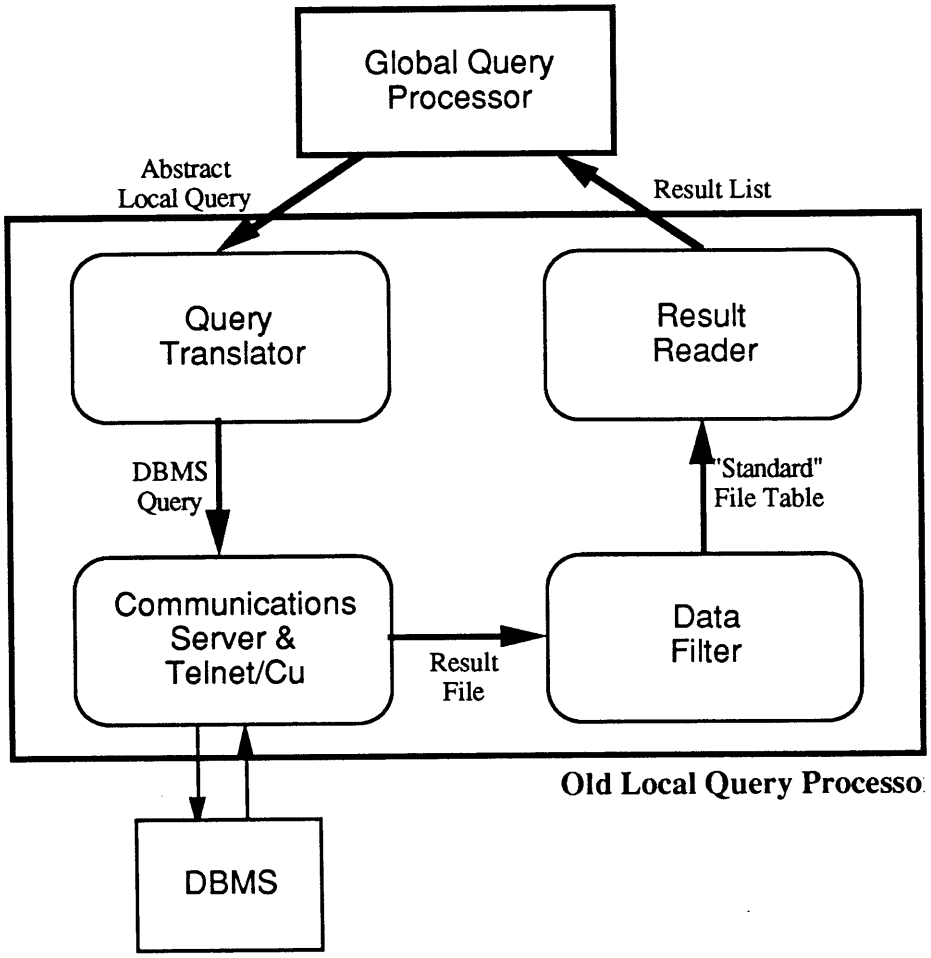


Figure 2: Schematic of the old layout of the LQP

2.2 Changing the LQP Structure

Of the different components of the LQP above, the communications server is the most suitable candidate for modularity. This is because the basic function of the communications server login to a host, issue a query and retrieve data. In most cases, this is done by sending a command at the right time (in most cases, when a

prompt is received) from the moment a connection is made till logoff. Although each of the other components are modularizable, the task is not as easy and may not be worth the time and effort. This thesis will address only the issue of the communications server. A modular communications server would change the LQP structure slightly as shown in Figure 3 below. This change in structure will make future plans to multitask CIS/TK LQP's or even multiplexing the communications server much easier. The modular communications server will essentially be data-driven[7].

[7] Act according to the data which it receives as opposed to having hard coded conditions.

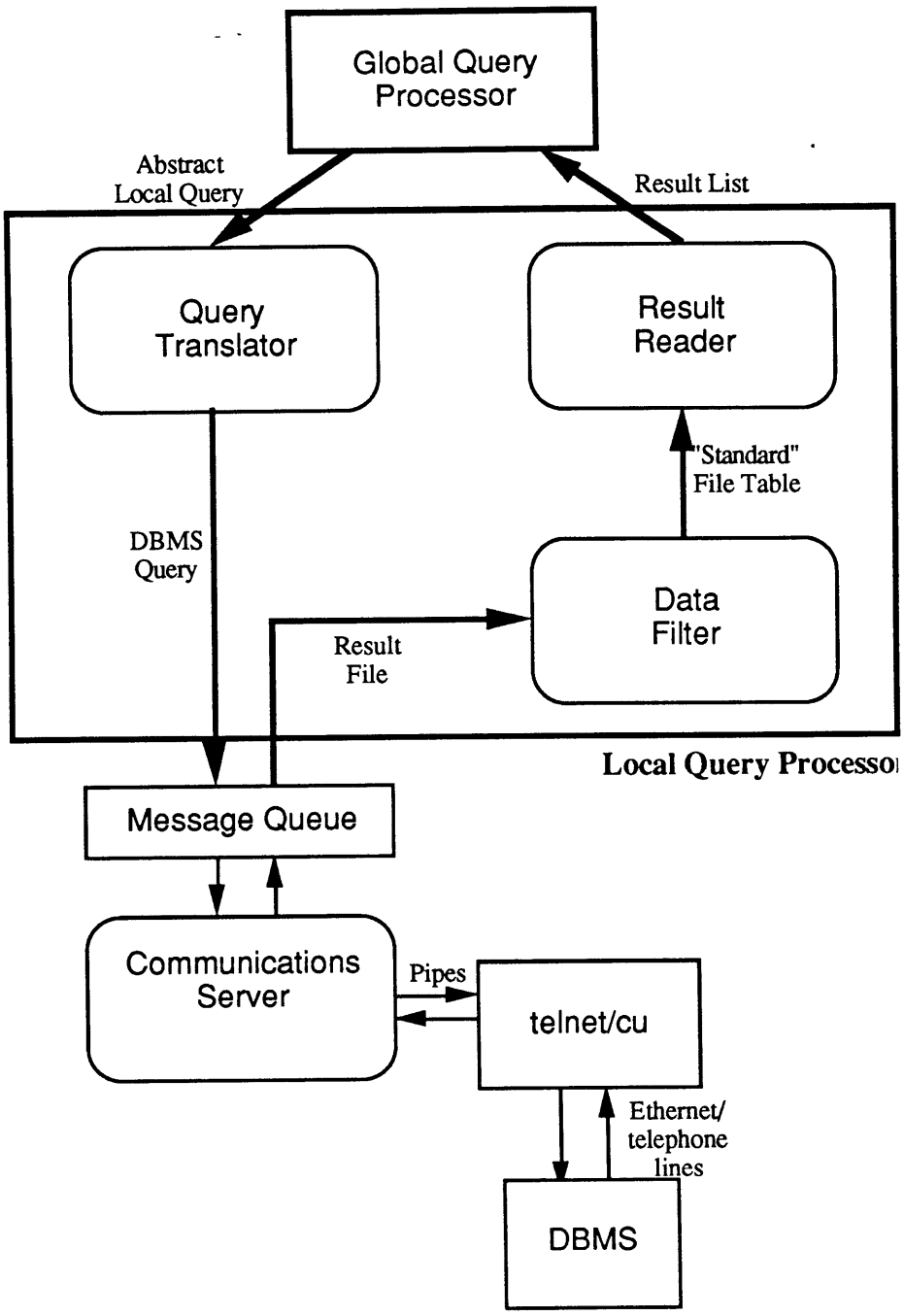


Figure 3: Schematic of the newly proposed layout of the LQP

Besides having to separate the communications server from the LQP's, the communications server itself needs some changes because it has some shortcomings that will affect performance and sometimes the robustness of the system.

2.3 Shortcomings of the Existing CS

The first shortcoming of the current version of the communications server is that it behaves like a "blind typist"; i.e. it assumes that perfect login conditions always hold. However, in the real world, this is not true. A foreign host (database server) that is heavily used may take longer to respond to a connect or to prompt for a password. One can see that the procedure currently used to connect to host machines, by echoing data at specific fixed time durations, is not robust in many instances. It is highly likely that the password or login id is echoed to the foreign host before it is prompted for. The communications server could potentially send the database commands before the database program on the foreign host is loaded. If a file does not exist or if some host conditions change, the communications server will not know at all and continue going through its fixed procedures until the result reader or the human being at the terminal realizes that something is amiss.

The second shortcoming is that the current communications server logs in and out for each database request. Keeping in mind that the login procedure takes a significant amount of time (even when it is successful), this turns out to be quite inefficient if a user is making a more than one query within a short amount of time.

2.4 CS Solutions and Game Plan

The new communications server will also include a new feature or enhancement that may become the first step towards making CIS/TK more efficient by using a method of software multiplexing, i.e., enabling the system to make multiple queries to different (or the same where possible) machines at the same time and thus use the distributed retrieval power simultaneously without

having to wait for one query to be completed before starting another. The existing CIS/TK design, particularly the LQP's, cannot handle multiplexing. However, the communications server will provide part of the mechanisms necessary to support multiplexing in the future.

The communications server will be implemented in two stages. The first stage will involve implementing a communications server that at least does what the existing communications server in each LQP does with one enhancement: a time-out feature. The time-out feature is especially necessary to handle the case of the foreign host freezing or if the foreign host never issues the expected prompt. This first stage essentially addresses the first shortcoming of the existing communications server mentioned earlier while the next stage addresses the second shortcoming and the additional enhancement. The second stage will involve implementing a multiplexing version of the communications server. This approach is used because of the the work in the second stage cannot really be tested because CIS/TK currently does not multitask its LQP's. The "multiplexing" version of the communications server will be not relogin if the connection has already been established previously and not terminated. The code for both the versions are in Appendix A.

The communications server also has to be able to handle the different communications protocols needed to connect to different machines for encapsulating the machine idiosyncrasies. For example, to communicate with the AT&T 3B2, each message line should be transmitted full-duplex and terminated with a New Line Character. On, the other hand, the I.P. Sharp service requires message lines transmitted half-duplex terminated with a Carriage Return/New Line sequence. The Reuters service needs message lines to be transmitted full-duplex terminated with just a Carriage Return character.

The second part of this thesis puts the communications server to the test through the implementation of an LQP for the I.P Sharp Disclosure database. In addition, the I.P. Sharp LQP is a good test for the new communications server because unlike existing supported systems, I.P. Sharp is not accessible through Ethernet using telnet[8]. Instead, the communications server will use cu[9]. As mentioned earlier, I.P. Sharp also requires a different communications protocol.

The implementation of this LQP would take CIS/TK one step further in terms of logical connectivity. It would also show that the existing AQP and GQP are modular enough to take on an LQP of a different nature from existing LQP's because of two reason. Firstly, unlike existing supported DBMS's, the I.P. Sharp service does not use SQL as its query language but instead it's own query language that is more constrained and less powerful than SQL. Thus, the LQP would have to be able to make the necessary query translations to overcome this problem. Secondly, the I.P. Sharp LQP is written in C although all the other LQP's were written in LISP.

[8] the communications software on UNIX that enables connections to other machines on ethernet
[9] another communications package on UNIX that enables a user to use a system modem to establish a connection to a foreign host

3. Communications Server

The discussion in this chapter will be based on the first stage implementation of the communications server. The second stage is just an extension of the first, and will be discussed at the end of the chapter. The first stage implementation of the communications server should be used by LQP that do not require dynamic or interaction with the foreign host. On the other hand, the stage two implementation allows an LQP to interact in real time with the foreign host.

3.1 The C programming language

The C programming language was the programming language of choice for this task because of the low level system interaction needed to provide time-out routines and pipe and file handling. C's powerful string parsing and character handling capabilities were also required to process the output of the foreign host. Using C for this task was not difficult because CIS/TK runs on a UNIX system where C is the mother tongue of most system applications/utilities.

3.2 The directly called version (Stage One)

To solve the problem of the "blind" communications server, the communications server will react to the host prompts and its reaction depends on the data files generated by each LQP. The data driven design for the communications server helps make the communications server modular. Using a time-out function for reading files/pipes, the communications server will know if the login fails or if a database request is unsuccessful. The communications server will receive the host's echoes through named pipes which will be declared using the mknod command in the following shell commands:

mknod tocomm p

mknod fromcomm p

telnet host < tocomm > fromcomm &

Telnet (or cu) will run in the background and will be communicated with through the pipes. Figure 4 below shows the pipes in relation to the different modules of the system.

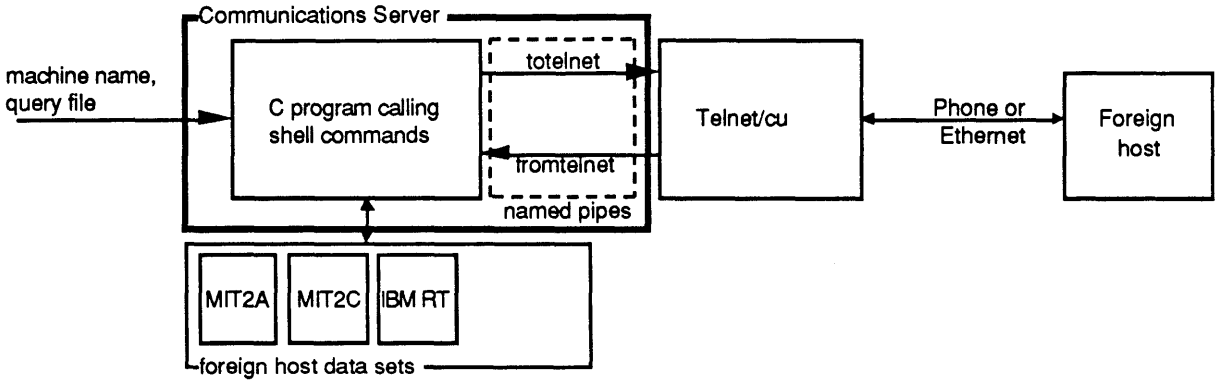


Figure 4: Usage of UNIX named within the CIS/TK system pipes

In the case where cu is used as the communications software, the first response has to be separated by at least a second and a half. This is because cu spawns a child process to perform the transmitting of data. If the communications server responds immediately upon receipt of the prompt, cu may not have started the transmission child quickly enough to send the response thus causing a lost response.

The communications server, commserv will then open, close, read and write to the named pipes: totelnet and fromtelnet as though they were regular files, using fopen(), fclose(), fprintf(), getc(), etc. The pipes have to be unwriteprotected to all users for the communications server to work properly. However, to make sure things work even better, the communications server actually removes the pipes

and recreates again. UNIX provides a system alarm, SIGALRM and a timer that will be used to implement the time-out feature.

A logfile, named logfile, always exists in /usr/cistk/demo/v2/lqp/dev is always appended to each session and can be viewed for debugging or error tracking purposes. The logfile will be created on a per session basis and will be deleted at the beginning of each new session.

The syntax of the data file that is currently used by the communication server for connecting to each foreign host is as follows:

```
commandline
prompt1, response1, timeout1
prompt2, response2, timeout2
          .
          .
          .
promptN, responseN, timeoutN
```

Each command in the prompt/response pair for the connect procedure has time-out argument that will cause the communications server to abort the session if a prompt is not received within the specified time. Each column is separated by a tab thus enabling the communications server to search for strings that contain spaces.

Optimally, this file should already exist given that most of the parameters are constant over time such as the temp files, login id, password, etc for any specific machine. The query should be created by the LQP and stored in a file that can be retrieved by commserv using a method mentioned below. However, for the first cut implementation, a shell between connect.lsp and commserv will be used as a sort of "compatibility mode". This shell can be found in Appendix B under 3b2fetch.c and RTFETCHFG.c. A cookbook method of creating such a datafile for the communications server is included in Appendix C.

In the future, each LQP can be modified to interface with the communications server directly but at that point in time, Stage Two of the communications server would be what the LQP's should support.

It is most likely that the first two prompts and responses will be for login and password although it is not limited to that. Response can take on a few different forms to serve different purposes. It uses the following syntax:

```
[*|%]response
```

With the '*' as the first character of responseX, response will be the name of a file that contains a query or just any response that may be dynamic over a period of time.

With the '%' option, keywords can be used to trigger certain functions. For example, the ability to toggle result capture can be done by using keywords like %capture and %endcapture. With the exception of %endcapture, any word following a '%' will be the name of file which whatever is received will be stored in. %endcapture will terminate the capturing the stream of data into the file. For both the '%' cases, the prompt is disregarded.

The timeout value on each line is in seconds and will cause the communications server to exit with an error message if it is reached. A default timeout value was considered but it was felt that having the creator of the datafile know to a good estimate for the timeout and put it into the datafile.

Due to the fact that some systems require Carriage Returns at a certain prompt, the capability to send a carriage return to such a system is to use "^M" in the position of a response in the data file.

A sample datafile created by the I.P. Sharp LQP for the I.P. Sharp/Disclosure database is as follows:

```
cu -s1200 -t 97237165 < tocomm 2> fromcomm &
[CR]          )                          100
[BEL]         )1769739:SLOAN             100
[BEL]         )LOAD 39 MAGIC             100
[BEL]         WIDTH 300                  100
[BEL]         DISPLAY '(CO EQ HONDA MOTOR CO) OR (CO EQ Honda Motor Co)' ADISCLOSURE
'COMPNO'      100
1 FOUND. PROCEED? (Y/N): [BEL]           Y      100
a             %ipresults                 100
[BEL]         ^M                         100
a             %endcapture                100
[BEL]         DISPLAY '(CO EQ HONDA MOTOR CO) OR (CO EQ Honda Motor Co)' ADISCLOSURE
'CO'          100
1 FOUND. PROCEED? (Y/N): [BEL]           Y      100
a             %ipresults                 100
[BEL]         ^M                         100
a             %endcapture                100
[BEL]         YEARLY DATED 85 TO 89      100
[BEL]         COLWIDTH 15                100
[BEL]         PUT '(CO EQ HONDA MOTOR CO) OR (CO EQ Honda Motor Co)' ADISCLOSURE
'CF,NI,NS'    100
1 FOUND. PROCEED? (Y/N): [BEL]           Y      100
[BEL]         'IBMPC'TABLE ABOVE         100
a             %ipresults                 100
[BEL]         ^M                         100
a             %endcapture                100
[BEL]         )OFF                       100
Disconnected  ^M                         100
```

3.3 The indirectly called version (Stage Two)

Although the benefits of having a multiplexing communications server, the amount of work involved and the potential problems that have to be overcome make that option less attractive. Among the problems are: having to keep track of the named pipes for each LQP, having to keep track of the incoming data and prompts from each foreign host. A simpler and more feasible solution is to have "simulated" multiplexing, explained below.

For stage 2, using message queues, the communications server does not have to provide the multiplexing capability. Instead, each of the LQP's will have to be modified so that they check for messages in the message queue every once so often if any particular LQP isn't the only one that is active. The modification to

the LQP's will involve attaching an interface (probably best written in C for ease of accessing and handling message queues and pipes). This interface will have to be able to read and write to the message queue by looking for specific message ids via named pipes explained in the previous section. As far as the communications server is concerned, one copy of it will be run in the background for every LQP that needs to connect to a foreign host. The overall big picture is as follows:

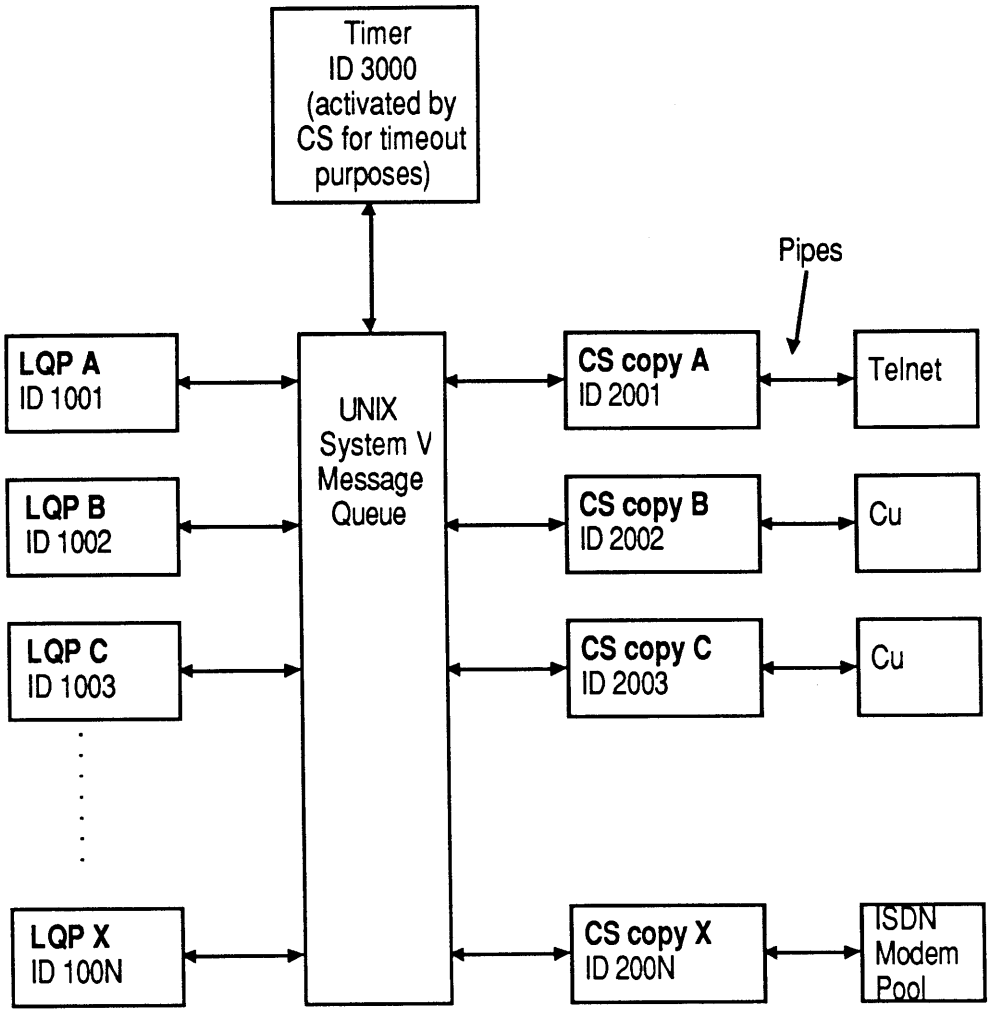


Figure 5: Overview of the Stage Two implementation of the 'simulated multitasking communications server

Each LQP will send commands to the Communications Server using the UNIX System V message queueing feature. Using the message queueing feature is a very attractive solution because currently it enables the simulation of a multiplexing communications server. In the future, it can become a central repository for messages and potentially enable more modular components of the CIS/TK to be developed through the use of a more standard message/parameter passing interface.

The commands which the communications server will accept is as follows:

```
connect machine duration
```

The machine argument is used by the Communications Server to identify which data file should be used to perform the initial connection and setup to use the respective databases. The duration argument is sent by the Communications Server to the timer process to automatically exit or break the connection at the end of the duration. The format for the connection data files is the same as that for the communications server in Stage One of the implementation. If the connection process is successful, the Communications Server will return a message, "Connected". The timeout value will be relayed by the Communications Server to the timer process with an error message containing the last message response sent that is to be sent to the calling LQP upon time-out, i.e. when the connection process fails.

The Communications Server will only perform the connect procedure if there isn't already a connection to the particular machine, i.e., if a connection was made previously for a query and not disconnected (logged out) and the last duration has not yet expired. However, the new duration will be put in effect.

Currently, this feature has not been tested yet due to inadequate time and testing facilities.

This file could potentially contain the query and the necessary commands to retrieve data that can be obtained in a straightforward manner. For example, the connect file for the Informix database on the MIT2C would look like this:

```
telnet mit2c < tocomm > fromcomm &
login:  demo     100
Password:        cis/tk  100
is:      h19     100
$        cd /usr/pagetm/cis      100
$        UNLOAD TO /usr/tmp/results.tmp SELECT CO, COMPNO FROM
GENINFO WHERE COMPNO = 2530 | isql -
$        %/usr/cistk/demo/lqp/v1/connect1.tmp    100
$        cat /usr/tmp/results.tmp        100
$        %endcapture     100
$        logout  100
```

This is quite similar in format to the datafile used by the Stage One communications server.

The Communications Server essentially translates each line in the data file to the following command lines which can also be issued stand-alone by the LQP to the Communications Server through the message queues.

```
listen prompt timeout [-]
```

The prompt argument is what the Communications Server looks for from the foreign machine. If the prompt is found, listen returns a 'found' message to the sender. The time-out argument will be relayed by the Communications Server to the timer process with a time-out error message that is to be sent to the calling LQP upon time-out, i.e. when the expected prompt does not show. The '-' option switches off the capture to result file option and only works if the given prompt is found. The '-' option will also append an end marker to the result file.

```
answer response [+file]
```

Answer allows the LQP to send a response to the Communications Server to be relayed to the foreign host. The +file option switches on the capture mode. Any data or text from the foreign host will be captured into a file. The filename is specified after the '+' as designated by 'file' above. A beginning marker is attached to the beginning of the file before the data incoming data from the foreign host. The beginning marker is then used by the respecitive LQP's for parsing purposes. The capture mode ends when the listen command described above is issued with the '-'\option.

```
exit
```

This command basically logs off the foreign machine, removes the pipes and closes the result files.

As mentioned earlier, this version of the communications server cannot be tested under the CIS/TK system because of the limitation of the CIS/TK LQP's to multitask and the non-existance of the LQP's to read from, write to and process the data from the message queues. However, this version runs stand alone for any given foreign host. Code is included, although currently commented out to read the message queue. For every different copy that has to be run, the queue ID has to be unique.

3.4 Pathname Independence

The communications server and the LQP's should optimally call and use files which are path independent. Thus a system of passing the necessary pathnames is needed. There are three possible methods to do this:

First, string constants can be used in each source file so that if the system moves, only a few constants at the top of each file has to be modified. The problem with this method is that if there are many files, then this may be quite a tedious process.

The second possibility is to use system variables to store strings like lqppath, devpath etc. In this case, each program would have to call a routine to read thos system variables at the beginning of each file. The limitation to this method would be that if the method of storing the pathnames changed, all the source files would have to have the function removed or mofified.

The third possibility is to have the pathnames passed down from the LQP or even the GQP in terms of objects stored in LISP. This seems to be the best possibility. However, due to time constraints and because many sections of the CIS/TK system would have to be modified, this will have to wait. Currently, there are two strings in each file called lqppath and devpath that are initialized at the beginning of each file. When the pathname passing is implemented, these strings can just be initialized with the arguments passed to each program.

4. I.P. Sharp LQP

This chapter discusses the implementation of the I.P. Sharp Disclosure LQP. This LQP is written in C unlike the rest of the existing LQP's and those under development. C was a suitable language to use for this LQP because a lot of the processing in this LQP pertained to parsing queries and data files. Using C for the entire LQP made interfacing each of the different function modules much easier. Modifying the LQP in the future or tracing through its logic will be much easier than if different sections/modules of the LQP were written in different languages[10].

4.1 I.P. Sharp Background

The I.P. Sharp database service is a comercial on-line database based in Toronto, Canada. The I.P. Sharp service offers two databases of interest to CIS/TK: Disclosure and Currency. The LQP described in this chapter will support Disclosure although support for Currency can easily be added in the future when necessary. The Disclosure database contains financial information obtained from financial reports of more than 12,000 companies that report to the Securities Exchange Commision in the United States. Most of these companies are incorporated in the U.S. although there are a number of non-U.S. companies included in the database.

4.2 I.P. Sharp LQP Design

This section discusses the structure of the I.P. Sharp LQP and the strategy used. Where perfect solutions were not possible, the tradeoffs are discussed. The

[10] For example, the LQPs for the Informix and Oracle databases

comments in the code for this LQP can be used as a supplement and may give a more genaral view of what each section of the code does. The code for the I.P. Sharp LQP can be found in Appendix A.

I.P. Sharp databases can be either menu driven or prompt/command driven. Therefore, there are two designs in which the I.P. Sharp LQP can be implemented. The menu driven interface is very suitable for users who are actually logging on in person because it gives a choice of options at any given time. The command driven option, on the other hand, requires knowledge of the database system, its commands and query language. However, the command driven method of retrieving information is much more efficient and faster. Since all the knowledge of the query language, databases and data fields can be incorporated into the LQP, there is nothing to be gained by using the menu driven option. In fact, the menu options are more likely to change often over time as opposed to the command option. Under the command option, even if things are added existing column names and query structure stored in the LQP would still be usable. The menu driven option, which sends more prompts over the phone line is more like to encounter errors and thus cause the communications server to miss prompts. For the above reasons, the I.P. Sharp LQP will utilize the command option of the service.

The I.P. Sharp LQP follows the newly proposed LQP structure with the communications server pulled out of the LQP itself. In trying to make the LQP as modular as possible, existing modules from the existing LQP's were used. For example, the Abstract Query Language (AQL) to SQL translator (sql.lsp) from the Informix and Oracle machines' LQP's were used as the front end to the I.P. Sharp LQP. It was much easier to translate from SQL to I.P. Sharp query language than it would have been from the LISP lists. For the back-end of the I.P. Sharp LQP,

the file reader (filter.c and read.lsp) were used. The I.P. Sharp LQP can be set to debug mode by specifying the debug variable at the top of the file to be 1 (0 if debug off). This can later be replaced by a function that sets it depending on an object value in a CIS/TK slot.

The actual function names and their function is as pictured below in Figure 6.

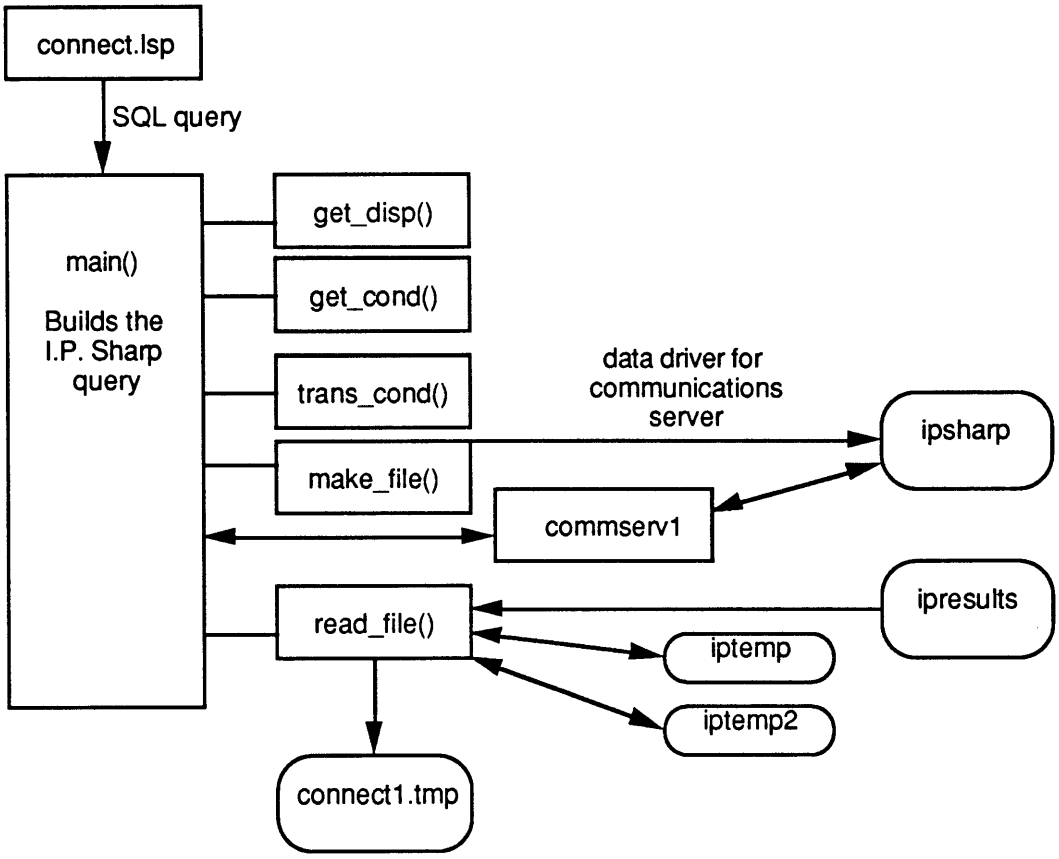


Figure 6: Flow diagram for the different modules in the I.P. Sharp LQP

The main function essentially reads a query in SQL form. It then proceeds to parse out the display column names and store them into an array using getdisp(). Getdisp() uses two other functions, searche() and searchb(), to search for the end of "SELECT" and the beginning of "FROM" respectively from the SQL query that

is passed to it. It then takes the substring in between and parses out the column names using ',' as the delimiter.

Some limitations in the I.P. Sharp system have caused some problems. One of the most major problems is that of I.P. Sharp not being able to handle a query that requests static data and time series data. For example, a single query cannot ask for both the company name (static data) and the annual earnings (time-series data) for the last five years for a given company number. Therefore, the LQP has to be able to separate the static data columns from the time-series data columns. All the display columns are then checked by check_disp() for validity in terms of whether they are currently supported as well as assigning a 'T' or 'S' flag to the corresponding flag array for time-series or static data respectively. Each column name that is passed to the function is checked against an array of structures that is currently hardcoded although at some point this could easily be read from a file or even passed down from the GQP[11]. Currently, the hardcoded column names is exactly what the AQP supports. Determining whether the column contains static or time-series data is necessary because separate queries for each of the two types of data (a solution to I.P. Sharp's limitation of not being able to handle both types of data in a single query).

This limitation of I.P. Sharp should be addressed at the GQP or the AQP level where time-series data and static data should be treated as coming as coming from two separate tables. As it stands, this LQP implementation for the I.P. Sharp system does some minor data joins when reconciling the results data from the queries. It is recommended that the GQP treat the static and time-series data

[11] when a method of passing an array or structure of data from IBCL to another program(C) is found

as though they are coming from two separate tables and make two separate queries. That way, the I.P. Sharp LQP will be much simpler in design and a lot of processing load will be taken of it. A list of the static and time-series columns available in the disclosure table is included in Appendix D under the sample run for get-columns.

The conditions in the SQL query are then parsed out and stored into an array using getcond(). Getcond() works almost in the same manner as getdisp() in the sense that it searches for the beginning of the condition section within the query and then looks for the beginning of certain keywords such as "OR", "AND" and "," for each of the individual conditions. Any additional keywords that may separate conditions in the future[12] can easily be added to the if statement.

Each of the conditions in the arrays are then translated into I.P. Sharp query language format using trans_cond() and then concatenated into a string, cond[]. Trans_cond() makes the following translations:

- '=' into 'EQ'
- "" into nothing because the I.P. Sharp Query Language does not require quotes around string literals
- puts a ')' before and a '(' after and 'AND' or 'OR'.

To be on the safe side, the cond[] string is always enclosed in a pair of parenthesis. The order of the conditions is kept the same during concatenation.

The rest of the main() program constructs the queries. Although all the static data could be retrieved in one query, there would be no way to separate the data

[12] like "NOT"

when it comes back because I.P. Sharp does not use any delimiters and spaces are not a valid delimiter because some data may contain more than one word and thus have spaces that are by no means delimiters. Unlike time-series data that can be retrieved in tabular form with fixed widths, static data cannot be retrieved as such. Therefore, one query is constructed for each of the static data columns requested and one query for all of the time series data. An array is used to keep track of which query each of the display columns are in. All of these queries are stored in an array of queries which is then passed on to the make_file() function that builds the data file for the communications server.

The data file is named `ipsharp`. The prompt that is looked for in most cases is a BEL character or a Control-G(ASCII 007). In this one case of looking for the correct prompts, I.P. Sharp was a difficult host to deal with. The BEL character is always sent as an error message and comes through stderr. This makes it difficult to test for the "n columns found. Proceed (Y/N)", which would have been more robust, because the BEL did not consistently come at the end of that prompt due to timing problems especially when the local (MIT2E) machine was busy.

The I.P. Sharp service is consistent in returning prompts in the correct order. If anything is amiss, the communications sever will timeout within the prespecified time. There is currently no capability in the GQP to tell the LQP whether the time-series data should be yearly dated or monthly dated. It could conceivably be included in the query in which case, it should not be a problem to add write another function to parse out the time-period into an array and then specify for that array to be written to the data file by the make_file() function.

The default for time series data is yearly dated from 1983 to 1989. This can be overriden by specifying a where clause for CF within a range of dates formatted as 8 digit integers. The "PUT" command is used to store the time-series data in a buffer space on I.P. Sharp. Together with the "WIDTH" command, the time-series data can be formatted into a table with the prespecified widths to enable easier parsing of the result file later on.

One undocumented feature of the I.P. Sharp service was discovered by trial and error that enables time-series data to be placed in a simple table without the date/year column on the right. The actual year or date should be included in the query by specifying a field called "CF" as mentioned earlier. The dates, being in numerical form, have embedded ',' and should be converted by the GQP. The command used is `'IBMPC'TABLE ABOVE`. This way, the column labels appear at the bottom after all the time-series data and the time-period labels are omitted from the left side of the table; things which make parsing the data file easier but could not be done using all the documented methods. A sample of the I.P. Sharp data file for the communications server can be found in Chapter 3.2.

For time-periods or in this case, years, for which there is no data, only blank lines will appear. This algorithm and the script for the communications server created by make_file assumes almost perfect login conditions[13]. This is a sufficient assumption because if there is anything wrong, either the timeout would catch it after having waited a certain amount of time or the read_file function later would detect a discrepancy in the data that was received if any. The logfile would further tell what the problem was. The directory in which the

[13] Initial trials have shown 100% success.

logfile, the communications server script and the results is in is specified by the variable lqppath which currently is set to "/usr/cistk/demo/v2/lqp/ipsharp". The communications server's path is specified by the string, devpath which is initialized to be "/usr/cistk/demo/v2/lqp/dev".

The above algorithm could be improved with an implementation of Stage Two of the communications server where the LQP could respond accordingly depending on whether data exists, or whether if there us too much data or even if something unexpected arises.

```
IPSharp LQP - Query Received:SELECT COMPNO, CO, CF, NI, NS FROM ADISCLOSURE
WHERE CO = 'HONDA MOTOR CO'
COMPNO S
CO S
CF T
NI T
NS T
The above are column names parsed out of the SQL query by the I.P. Sharp
LQP and typed static or time-series
(CO  EQ HONDA MOTOR CO )
IPSharp LQP - Queries generated:
DISPLAY '(CO  EQ HONDA MOTOR CO )' ADISCLOSURE 'COMPNO'
DISPLAY '(CO  EQ HONDA MOTOR CO )' ADISCLOSURE 'CO'
PUT '(CO  EQ HONDA MOTOR CO )' ADISCLOSURE 'CF,NI,NS'
rm: /usr/cistk/demo/v2/lqp/dev/ipresults non-existent
Finished reading file
Connected


)

 GHBGSBHSGBSBGBSGBM4M4M4M4MNMNMNMNM58585858HEHEHEHEH)1769739:SLOAN
These junk characters are actually printed over each other

  6230) 1989-05-08 00:52:31 IPSA

SHARP APL SERVICE

        )LOAD 39 MAGIC

SAVED  1989-04-05 15:50:11

        WIDTH 300
```

```
      DISPLAY '(CO  EQ HONDA MOTOR CO )' ADISCLOSURE 'COMPNO'

1 FOUND. PROCEED? (Y/N): Y



3842                              answer
      DISPLAY '(CO  EQ HONDA MOTOR CO )' ADISCLOSURE 'CO'

1 FOUND. PROCEED? (Y/N): Y



HONDA MOTOR CO LTD                answer
        YEARLY DATED 85 TO 89

        COLWIDTH 15

         PUT '(CO   EQ HONDA MOTOR CO )' ADISCLOSURE 'CF,NI,NS'

1 FOUND. PROCEED? (Y/N): Y

         'IBMPC'TABLE ABOVE



     19,870,228          83,689       2,868,305

     19,880,331          56,676       1,584,622



          col 1          col 2          col 3
The above 3 lines are answers to the query.  Note that the years for which
there are no data are blank.

)OFF

  6230  1989-05-08 00:53:15 IPS
```

```
CONNECTED      00:00:44  TO DATE      01:07:41

CPU UNITS        197.403  TO DATE      1398.113

KILOCHARS          0.835  TO DATE         5.431



Lost Carrier


Disconnected
```

Figure 7: Sample I.P. Sharp Session (With non-printing ASCII characters hidden)Communication server's responses are in bold and comments are in italics

```
[NL]IPSharp LQP - Query Received:SELECT COMPNO, CO, CF, NI, NS FROM
ADISCLOSURE WHERE CO = 'HONDA MOTOR CO'
[NL]COMPNO S
[NL]CO S
[NL]CF T
[NL]NI T
[NL]NS T
[NL](CO  EQ HONDA MOTOR CO )
[NL]IPSharp LQP - Queries generated:
[NL]DISPLAY '(CO  EQ HONDA MOTOR CO )' ADISCLOSURE 'COMPNO'
[NL]DISPLAY '(CO  EQ HONDA MOTOR CO )' ADISCLOSURE 'CO'
[NL]PUT '(CO  EQ HONDA MOTOR CO )' ADISCLOSURE 'CF,NI,NS'
[NL]rm: /usr/cistk/demo/v2/lqp/dev/ipresults non-existent
[NL]Finished reading file
[NL]Connected[BEL]

[NL]

[NL])
[NL]

[NL]
GHBGSBHSGBSBGBSGB[BS][BS][BS][BS][BS][BS][BS][BS][BS][BS][BS][BS][BS][B
S][BS][BS]M4M4M4M4MNMNMNMNM[BS][BS][BS][BS][BS][BS][BS][BS][BS][BS][BS]
[BS][BS][BS][BS][BS]58585858HEHEHEHEH[BS][BS][BS][BS][BS][BS][BS][BS][BS][B
S][BS][BS][BS][BS][BS][BS][BS][BS][BS][BEL])1769739:SLOAN
[NL]

[NL]  6230) 1989-05-08 00:52:31 IPSA

[NL]
[NL]SHARP APL SERVICE
[NL]

[NL]      [BEL])LOAD 39 MAGIC
```

```
[NL]

[NL]SAVED  1989-04-05 15:50:11

[NL]       [BEL]WIDTH 300
[NL]

[NL]       [BEL]DISPLAY '(CO  EQ HONDA MOTOR CO )' ADISCLOSURE 'COMPNO'
[NL]

[NL]1 FOUND. PROCEED? (Y/N): [BEL]Y
[NL]

[NL]                [BS] [BS] [BS] [BS][BEL]
[NL]

[NL]3842

[NL]       [BEL]DISPLAY '(CO  EQ HONDA MOTOR CO )' ADISCLOSURE 'CO'
[NL]

[NL]1 FOUND. PROCEED? (Y/N): [BEL]Y
[NL]

[NL]                [BS] [BS] [BS] [BS][BEL]
[NL]

[NL]HONDA MOTOR CO LTD

[NL]       [BEL]YEARLY DATED 85 TO 89
[NL]

[NL]       [BEL]COLWIDTH 15
[NL]

[NL]       [BEL]PUT '(CO  EQ HONDA MOTOR CO )' ADISCLOSURE 'CF,NI,NS'
[NL]

[NL]1 FOUND. PROCEED? (Y/N): [BEL]Y
[NL]

[NL]       [BEL]'IBMPC'TABLE ABOVE
[NL]

[NL]
 [BEL]
[NL]

[NL]     19,870,228         83,689      2,868,305

[NL]     19,880,331         56,676      1,584,622

[NL]

[NL]

[NL]          col 1          col 2          col 3
```

```
[BEL])OFF
[NL]       [BEL]

[NL]  6230  1989-05-08 00:53:15 IPS

[NL]CONNECTED      00:00:44  TO DATE      01:07:41

[NL]CPU UNITS       197.403  TO DATE      1398.113

[NL]KILOCHARS         0.835  TO DATE         5.431

[NL]
[NL]
[BEL]

[NL]Lost Carrier

[NL]

[NL]Disconnected[BEL]
```

Figure 8: Sample I.P. Sharp Session (With non-printing ASCII characters shown)

In the last part of main(), the old result file, `ipresults` is then removed before the communications server is invoked. When the communications server is done, the read_file() function processes the result file. Before read_file() converts the raw data in ipresults to formatted data in connect1.tmp, it removes the old(existing) copy of connect1.tmp so that the user is not mislead with old data if the current session had failed[14]. Read_file() reads the result file, counts the beginning and end markers to ensure that all the requested data has been returned and strips the beginning and end markers. An error message is printed if the number of beginning and end markers don't match. This will be the case if any of the queries has failed. Figure 9 shows a sample of the result file produced by the communications server. By keeping track of the beginning markers, read_file() is able to tell whether if the data between a pair of markers is supposed to be static or time-series data and thus can parse it accordingly.

[14] This is an improvement over the existing LQP's for the Informix machines and the Oracle machine where the data from the last query is passed back even if the current session fails.

Read_file() is also able to know which piece of data corresponds to which display column through the dispq[] array that had values assigned earlier by get_disp(). The static data is easy to extract because it essentially comes on one line[15]. The time series data is a bit more of a hassle because the data for more than one display column may appear on one line. However, because make_file() had earlier requested that the WIDTH be set to 15 (or any other prespecified number), read_file() can just parse data by number of characters including the spaces that normally separates the different columns of time-series data. The static and time-series data is then reassembled into a table format with the vertical bar '|' as the delimiter. This delimiter was chosen to try to create a file that the same format as the Informix unload[16] files. Figure 10 below shows a sample of a formatted result file.

```
*****BEGINNING MARKER*****

3842

      ********END MARKER********
*****BEGINNING MARKER*****

HONDA MOTOR CO LTD

      ********END MARKER********
*****BEGINNING MARKER*****

     19,870,228          83,689      2,868,305

     19,880,331          56,676      1,584,622



          col 1          col 2          col 3

********END MARKER********
```

[15] In this case, one line is defined as the space between two carriage return/line feed pairs.

[16] Informix's unload feature dumps the results for a given query into a UNIX file instead of out on the terminal.

Figure 9: Sample I.P. Sharp data returned in a file by the communications server.

```
*****BEGINNING MARKER*****
3842|HONDA MOTOR CO LTD|19,870,228|83,689|2,868,305|
3842|HONDA MOTOR CO LTD|19,880,331|56,676|1,584,622|
********END MARKER********
```

Figure 10: Sample I.P. Sharp data after being formatted

5. Conclusion

The new Communications Server has, through the use of some UNIX features, addressed some of the issues of physical connectivity and has at the same time solved some of the problems faced by the preceding version of the Communications Server. However, there is a lot more standardizing that can be done among the different LQP's to help make the Communications Server's job easier and more more efficient.

The I.P. Sharp LQP has taken on a lot of responsibility and is not as dumb as the Informix or Oracle LQP's. The next step to making the LQP more modular is to let either the AQP or GQP issue the necessary separate static data and time-series data queries and consolidating the data instead of having the LQP do it. As it stands, the major part of this LQP is splitting the query into the data queries and the time-series query and consolidating the data into the standard form.

Appendix D contains a sample run of CIS/TK using the I.P. Sharp LQP and the new communications server.

6. References

AT&T UNIX System V Programmers Reference Manual

I.P. Sharp Reference Manual, February 1988

Maria L. Paget, A Knowledge-Based Approach toward Integrating International On-line Databases, January 1989, Cambridge, MA

Y. Richard Wang & Stuart E. Madnick, Connectivity Among Informations Systems (CACM, Computing Practices), 1988, Cambridge, MA

Y. Richard Wang & Stuart E. Madnick, Logical Connectivity: Applications, Requirements, and An Architecture (Management Informations Systems Quaterly), 1988, Cambridge, MA

Y. Richard Wang, David C. Horton & Stuart E. Madnick, Inter-Database Instance Identification in Composite Information Systems, (Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences, January 1989), Cambridge, MA

Y. Richard Wang, David C. Horton, T.K. Wong & Stuart E. Madnick, Concept Agents In CIS/TK: a Tool Kit for Composite Information Systems (Proceedings of the International Computer Symposium 88, Taipei, Taiwan), 1988, Cambridge, MA

Appendix A: C Code

commserv1.c (direct connect version)

```
#include<stdio.h>
#include<stropts.h>
#include<malloc.h>
#include<signal.h>
#define MAXLINES 25
    FILE   *tocomm,      /* file pointer to pipe to cu/telnet     */
           *fromcomm,    /* file pointer to pipe from cu/telnet   */
           *fout;           /* file pointer to result file           */
    int    capture,      /* result capture toggle       */
           debug=0,      /* debug mode toggle       */
           timer=30;     /* default timeout(sec)        */
    char   outfile[50],  /* result filename          */
                  devpath[100],
                  lqppath[100],
                  filename[100];

main(argc,argv)
    int argc;
    char *argv[];
{
    FILE   *fp, *fopen();
    char   prompt[MAXLINES][100], response[MAXLINES][1000], telcom[150],
command[100];
            extern char devpath[], lqppath[], filename[];
    int    time[MAXLINES], x=0, y=0, status,
           endcapture;  /* result endcapture toggle*/
    unsigned alarm();
    void timeup();
    capture = 0;

    strcpy(devpath,"/usr/cistk/demo/v2/lqp/dev/");
    strcpy(lqppath,"/usr/cistk/demo/v2/lqp/ipsharp/");


    if (!(fp = fopen(argv[1],"r")))
    {
       printf("Comm-Server: Can't open LQP script file '%s'.\n", argv[1]);
       return(0);
    }
    fscanf(fp,"%[^\n]%*c",telcom);

    while (x < MAXLINES && fscanf(fp,"%[^\t] %[^\t] %d%*c",prompt[x], response[x],
&time[x])!=EOF)
    {

       if(strcmp(response[x],"^M")==0)response[x][0]=0;
           if(debug==1)printf("%s\t%s\t%d\n",prompt[x], response[x], time[x]);
       x++;
    }
    printf("Finished reading file\n");
    setbuf(stdout,0);
    sprintf(command,"rm %stocomm",devpath);
          system(command);
    sprintf(command,"rm %sfromcomm",devpath);
          system(command);
    sprintf(command,"/etc/mknod %stocomm p",devpath);
          system(command);
```

```
    sprintf(command,"chmod +w %stocomm",devpath);
          system(command);
    sprintf(command,"/etc/mknod %sfromcomm p",devpath);
          system(command);
    sprintf(command,"chmod +w %sfromcomm",devpath);
          system(command);

    signal(SIGALRM,timeup);    /* set alarm clock */
    alarm(timer);    /* set timer */
    system(telcom);
    sprintf(filename,"%stocomm",devpath);
    tocomm = fopen(filename,"w");
    setbuf(tocomm,0);
    sprintf(filename,"%sfromcomm",devpath);
    fromcomm = fopen(filename,"r");
    setbuf(fromcomm,0);

    do
    {
        if (strcmp(response[y],"%endcapture")==0)
        {
              endcapture = 1;
        }
        else
            if (response[y][0]=='%')
            {
                capture = 2;
                endcapture = 0;
                strcpy(outfile,response[y]+1);
            }
            else
            {
                status = lookfor(prompt[y],response[y],time[y]);
                if (status == 1 && capture == 2)
                {
                    capture = 1;
                    fout = fopen(outfile,"a");
                    fprintf(fout,"*****BEGINNING MARKER*****");
                    fclose(fout);
                }
                if (endcapture == 1)
                {
                   endcapture=0;
                    capture = 0;
                    fout = fopen(outfile,"a");
                    fprintf(fout,"********END MARKER********\n");
                    fclose(fout);
                }
            }
        y++;
    } while (y < x);
    fclose(tocomm);
    fclose(fromcomm);
    sprintf(command,"rm %stocomm",devpath);
      system(command);
    sprintf(command,"rm %sfromcomm",devpath);
      system(command);
    sprintf(command,"chmod +w %sipresults",lqppath);
    system(command);
}

lookfor(prompt,response,time)
char prompt[], response[];
int time;
```

```
{
    FILE *fp1, *log;
    char c, query[1000], *malloc();
    int i=0, t=0, nfds=1, poll();
          extern char filename[], devpath[];
    extern int capture;
    extern FILE *fout;

    alarm(time); /* set timer */

    do
    {
           c=getc(fromcomm);

           if(debug==1) printf("%c",c);
           sprintf(filename,"%slogfile",devpath);
           log = fopen(filename,"a");
           fprintf(log,"%c",c);
           fclose(log);
           if (capture == 1)
           {
              fout = fopen(outfile,"a");
              fprintf(fout,"%c",c);
              fclose(fout);
           }
           if(debug==1) printf("%d %d %d %d\n", c, prompt[i], i, strlen(prompt));
           if (c == prompt[i])
           {
              if (i == strlen(prompt)-1)
              {
                 alarm(timer);    /* reset timer */
                 if (response[0] == '*')
                 {
                    fp1 = fopen(response+1,"r");
                    fscanf(fp1,"%s",query);
                    printf("\n%s\n",query);
                    fprintf(tocomm,"%s\n",query);
                 }
                 else
                   {
                     system("sleep 2");
                      fprintf(tocomm,"%s\n",response);
                      printf("%s\n",response);
                   }
                 return(1);
              }
              else
                 i++;
           }
           else
              i = 0;
    } while (t < time);
    return(0);
}

void timeup()
{
    fprintf(stderr,"Timed out\n");
    exit(0);
}
```

commserv2.c (multiplexing version)

```
#include<stdio.h>
#include<malloc.h>
#include<signal.h>
#define MAXLINES 25     /* max # of lines per connect file      */
#define DIR "/usr/cistk/demo/v2/lqp/dev/"

    FILE   *tocomm,      /* file pointer to pipe to cu/telnet    */
           *fromcomm,    /* file pointer to pipe from cu/telnet  */
           *fout;           /* file pointer to result file          */
    int    capture,      /* result capture toggle      */
           debug=0,      /* debug mode toggle       */
           timer=30;     /* default timeout(sec)       */
    char   outfile[50],  /* result filename          */
                  devpath[100],
                  lqppath[100],
                  filename[100];

main(argc,argv)
    int argc;
    char *argv[];
{
    FILE   *fp, *fopen();

    char   prompt[MAXLINES][100], response[MAXLINES][1000], telcom[150],
command[100];
            extern char devpath[], lqppath[], filename[];

    int    time[MAXLINES], x=0, y=0, status,
           endcapture;  /* result endcapture toggle*/

    unsigned alarm();          /* for timeout routine            */
    void timeup();          /*                                */

    extern int capture;
    struct token
    {
       char byte[80];
    };
    struct token toktbl[10];
    capture=0;
    scanf("%[^\n]%*c",command); /* read from queue using fscanf and pipes */
    numtok = tokenize(command,toktbl);
    switch(toktbl[0].byte[0])
    {
       case 'c':
       {

/***********************************/
/* Routine for reading connect file  */
/***********************************/
    strcat(command,DIR);
    strcat(command,argv[1]);
    if  (!(fp=fopen(toktbl[1].byte,"r")))
    {
       printf("ERROR! Can't open file.\n");
       return(0);
    }

    fscanf(fp,"%[^\n]%*c",telcom);    /* reading initiation command */

    while (x < MAXLINES)          /* reading prompt, response and timeout sets */
    {
```

```
        fscanf(fp,"%[^\t] %[^\t] %d%*c",prompt[x], response[x], &time[x]);
        if(strcmp(response[x],"^M")==0)response[x][0]=0;
            if(debug==1)printf("%s\t%s\t%d\n",prompt[x], response[x], time[x]);
        x++;
    }
    printf("Finished reading file\n");

    setbuf(stdout,0);    /* Output buffer set to zero for real time pipe reads */

/*********************************************/
/* Removing old pipes and setting new ones    */
/*********************************************/
    sprintf(command,"rm %stocomm",devpath);
          system(command);
    sprintf(command,"rm %sfromcomm",devpath);
          system(command);
    sprintf(command,"/etc/mknod %stocomm p",devpath);
          system(command);
    sprintf(command,"chmod +w %stocomm",devpath);
          system(command);
    sprintf(command,"/etc/mknod %sfromcomm p",devpath);
          system(command);
    sprintf(command,"chmod +w %sfromcomm",devpath);
          system(command);

    signal(SIGALRM,timeup);    /* set alarm clock */
    alarm(timer);    /* set timer */

    system(telcom); /* invoking communications program and pipes */

/*********************************************/
/* Setting up file descriptors for pipes      */
/*********************************************/
    sprintf(filename,"%stocomm",devpath);
    tocomm = fopen(filename,"w");
    setbuf(tocomm,0);
    sprintf(filename,"%sfromcomm",devpath);
    fromcomm = fopen(filename,"r");

/*********************************************/
/* Looping through and parsing prompts        */
/*********************************************/
    do
    {
    /* testing for endcapture */
        if (strcmp(response[y],"%endcapture")==0)
        {
              endcapture = 1; /* Setting endcapture toggle on */
        }
        else
        /* testing for capture by looking for % */
            if (response[y][0]=='%')
            {
            /* set capture toggle standby, endcapture off */
            /* and parse out result filename          */
                capture = 2;
                endcapture = 0;
                strcpy(outfile,response[y]+1);
            }
            else
            {
            /* Call lookfor function to test pipe for prompt,          */
            /* respond with response if prompt is found within time    */
            /* and return status = 1 if successful                     */
```

```
                if((status = lookfor(prompt[y],time[y]))==1)
                    answer(response[y]);

                /* If prompt found, set capture toggle to on and put     */
                /* beginning toggle at the beginning of the .result file */
                /* (when lookfor is looking for the next prompt, the     */
                /* incoming data will automatically be stored into the   */
                /* result file)                                             */

                if (status == 1 && capture == 2)
                {
                    capture = 1;
                    fout = fopen(outfile,"a");
                    fprintf(fout,"*****BEGINNING MARKER*****");
                    fclose(fout);
                }

                /* If endcapture toggle is set to on, append end marker */
                /* on to result file and set endcapture and capture     */
                /* toggles to off                                    */

                if (endcapture == 1)
                {
                   endcapture=0;
                    capture = 0;
                    fout = fopen(outfile,"a");
                    fprintf(fout,"********END MARKER********\n");
                    fclose(fout);
                }
            }
        y++;
    } while (y < x);   /* while there are still prompts to be waited on */
                       /* from the connect file                            */
        }
        case 'l':
        {
            status=lookfor(prompt,time);
        }
        case 'a':
        {
            answer(response);
        }
        case 'e':
        {
        }
        default:
        {
        }
    }
}

    /*************************************************************/
    /* Close file descriptors for the pipes and remove the pipes */
    /*************************************************************/

    fclose(tocomm);
    fclose(fromcomm);
    sprintf(command,"rm  %stocomm",devpath);
     system(command);
    sprintf(command,"rm  %sfromcomm",devpath);
     system(command);
    sprintf(command,"chmod  +w  %sipresults",lqppath);
    system(command);
```

```
}


/*************************************************************/
/* Looffor routine that does the testing of incoming data    */
/* through the pipe until it is interrupted by SIGALRM at    */
/* timeout.                                                  */
/*************************************************************/

lookfor(prompt,response,time)
char prompt[], response[];
int time;
{
    FILE *fp1, *log;
    char c, query[500], *malloc();
    int i=0, t=0;
    extern int capture;
    extern FILE *fout;
    alarm(time);        /* set timer to new timeout value */
    do
    {
        c=getc(fromcomm);   /* read inpipe character by character      */
        if(debug==1)     /* output read character to terminal - not    */
        printf("%c",c);  /* necessary and for information purposes only */


            sprintf(filename,"%slogfile",devpath);
            log = fopen(filename,"a");
            fprintf(log,"%c",c);
            fclose(log);


        /* If capture toggle is on, then write data from pipe into  */
        /* result file                                              */

        if (capture == 1)
        {
            fout = fopen(outfile,"a");
            fprintf(fout,"%c",c);
            fclose(fout);
        }

/*************************************************/
/* Routine that tests incoming data for prompt */
/*************************************************/
        if (c == prompt[i])
        {
        /* If prompt found */
            if (i == strlen(prompt)-1)
            {
                alarm(timer);    /* reset timer so that lookfor is not   */
                                 /* interrupted when receiving incoming */
                                 /* data                                 */
                return(1);    /* return 1 for status in main for successful run */
            }
            else
                i++;   /* counter for comparing next character of prompt[] with */
                       /* incoming data from pipe                                */
        }
        else
            i = 0;
    } while (1);
}
```

```
/* routine run at timeout */

void timeup()
{
    fprintf(stderr,"Timed out!\n");
    exit(0);
}

answer(response)
char response[];
{
  FILE *fp1;
  char query[500];
             /* If first character of response is '*', then filename  */
             /* that contains response follows                        */

             if (response[0] == '*')
             {
                 fp1 = fopen(response+1,"r");
                 fscanf(fp1,"%s",query);          /* read response from file */
                 printf("\n%s\n",query);
                 fprintf(tocomm,"%s\n",query); /* write response to pipe   */
             }
             else
             {
                 system("sleep 2");  /* Currently hardcoded pause to */
                                     /* accomodate slowness of cu in */
                                     /* creating child transmission  */
                                     /* process                      */

                 fprintf(tocomm,"%s\n",response);/* write response to pipe   */
             }
}

tokenize (tline, toktbl)
char tline[80];
struct token
{
    char byte[80];      /* Each token = 80 bytes */
};
struct token toktbl[10];   /* Token table = 10 tokens */

{
    int i,j,a ;

    j=0;
    a = 0;

    for ( i = 0; i < 80 && a <= 9 ; i++)
    {
        if (tline[i] != ' ' && tline[i] != '\0')
       {
            toktbl[a].byte[j] = tline[i];
           j++;

            if (tline[i+1] == ' ' || tline[i+1] == '\0')
           {
               toktbl[a].byte[j] = '\0';
              a++;
              j=0;
           }
       }

        if (tline[i] == '\0')
```

```
            return(a);

    }
    return(-1);
}
```

qt.c (I.P. Sharp LQP)

```
#include<stdio.h>
#include<string.h>

#define MAXCOL 10              /* max # of columns           */
#define MAXCOND 10           /* max # of cond                */
#define MAXQUERY 10          /* max # of IPSharp queries   */
#define MAXCHAR 15           /* max width of columns    data   */
#define MAX_IP_FIELDS 25
   int    debug=1;             /* debug mode toggle          */

main(argc,argv)
   int argc;
   char *argv[];
{
   char sql[400],           /* original SQL query         */
      disp[MAXCOL][10],        /* display cols               */
      cond[500],            /* conditions                    */
      tcond[MAXCOND][50], /* individual conditions          */
      dispt[MAXCOL][2],        /* display type (s,t)         */
      dispq[MAXCOL][3],        /* display query #, pos          */
      query[MAXQUERY][200],/* ipsharp queries           */
      devpath[100],            /* Device pathname for commserv1, pipes */
      lqppath[100],            /* LQP pathname for ipsharp datafile and result files
*/
      command[150];            /* string for system commands     */

   int x=0, t=0, n=0, m=0,     /* generic counter variables  */
      nq=0,mm=0,
      tsfound=0,           /* flag for time-series data   */
      ncol;                /* # of disp cols                */

/*************************/
/* get query in SQL form  */
/*************************/
   strcpy(sql,argv[7]);
   strcpy(devpath,"/usr/cistk/demo/v2/lqp/dev/");
   strcpy(lqppath,"/usr/cistk/demo/v2/lqp/ipsharp/");
   if(debug==1) printf("IPSharp LQP - Query Received: %s\n",sql);

/*************************/
/* Testing for INFO TABLES */
/*************************/
if(strcmp(sql,"INFO TABLES")==0)
{
   strcpy(query[0],"LIST SERVICES");
   strcpy(disp[0],"TABLE");
   strcpy(disp[1],"DESC");
   ncol=2;
   nq=1;
}
else
   if(strncmp(sql,"INFO COLUMNS FOR",16)==0)
   {
      /* parse out table name */
      strcpy(query[0],"ADISCLOSURE 'LISTFACTS'");
      strcpy(disp[0],"NO");
      strcpy(disp[1],"MNEM");
      strcpy(disp[2],"DESC");
      ncol=3;
      nq=1;
   }
   else
```

```
{
    /*************************************************************************/
    /* parse display columns out of 'sql' into 'disp[]' using getdisp()
*/
    /* and validate them as static or time series in dispt[x] using checkdisp()*/
    /*************************************************************************/
    getdisp(sql, disp);
    while(disp[x][0]!=0)
    {
       x++;
    }
    for(x=0; disp[x][0] != 0; x++)
    {
       dispt[x][0]=check_disp(&disp[x][0]);
       if(debug==1) printf("%s %s\n",disp[x],dispt[x]);
    }
    ncol=x;          /* store # of cols */


    /*************************************************************************/
    /* parse conditions out of 'sql' into 'tcond[]' using getcond()
*/
    /*************************************************************************/
    getcond(sql, tcond);

    /*************************************************************************/
    /* translate each condition in tcond[] into IPSharp format and           */
    /* concatenate into cond                                                 */
    /*************************************************************************/
    for(x=0; tcond[x][0] != 0; x++)
    {
       trans_cond(&(tcond[x][0]));
       strcat(cond,tcond[x]);
       if(debug==1) printf("%s\n",cond);
    }

    /*************************************************************************/
    /* build one query for each piece of static data needed and store in the */
    /* nth query array                                                       */
    /*************************************************************************/
    for(t=0;t<ncol;t++)
    {
       if(dispt[t][0]=='S')
       {
          strcat(query[n],"DISPLAY '");
          strcat(query[n],cond);
          strcat(query[n],"' ");
          strcat(query[n],"ADISCLOSURE '");
          strcat(query[n],disp[t]);
          strcat(query[n],"'");
          n++;                          /* increment query #             */
          sprintf(dispq[t],"%d1",n); /* Assign query number to dispq  */
       }
    }


    /*************************************************************************/
    /* build one query for all the time-series data                          */
    /*************************************************************************/
    for(mm=0;mm<ncol;mm++)
    {
       if(dispt[mm][0]=='T')
       {
          tsfound=1;
```

```
            break;
        }
    }
    if(tsfound==1)
    {
        strcat(query[n],"PUT '");
        strcat(query[n],cond);
        if(query[n][strlen(query[n])-1]==',')
            query[n][strlen(query[n])-1]=0;
        strcat(query[n],"' ");
        strcat(query[n],"ADISCLOSURE '");
        for(t=0;t<ncol;t++)
        {
            if(dispt[t][0]=='T')
            {
                m++;      /* increment position # of disp col in query */
                strcat(query[n],disp[t]);
                strcat(query[n],",");

                /* This limits the # of queries and the # of cols per query to 10)
*/
                sprintf(dispq[t],"%d%d",n+1,m);
            }
        }

        /* strip the last comma in the column list */
        if(query[n][strlen(query[n])-1]==',')
            query[n][strlen(query[n])-1]=0;

        strcat(query[n],"'");

        n++;   /* nq=number of queries */
    } /* End of tsfound if */

    if(debug==1)
    {
        printf("IPSharp LQP - Queries generated:\n");
        for(t=0;query[t][0]!=0;t++)
            printf("%s\n",query[t]);
    }
    nq=n;
} /* End of else */

make_file(devpath,lqppath,query, nq);

sprintf(command,"rm %sipresults",lqppath);
system(command);                          /* Removing old result file       */
sprintf(command,"%scommserv1 %sipsharp",devpath,lqppath);
system(command);                          /* Invoking communications server */
read_file(lqppath,devpath,disp,dispt,dispq, nq, ncol);

sprintf(command,"chmod +w %sconnect1.tmp",lqppath);
system(command);

} /* End of main() */

/*************************************************************************/
/* Function to pick out display column names from SQL query              */
/*************************************************************************/
getdisp(sql,disp)
char sql[], disp[][10];
{
    int x, y, s, m=0, n=0;
```

```
    x = searche(sql,"SELECT") + 1;    /* Find beginning of display col list   */
    y = searchb(sql,"FROM");          /* Find end of display col list         */

    /*************************************************************************/
    /* parsing routine to separate each of the col names and store in an array */
    /*************************************************************************/
    for(s=x; s<y; s++)
    {
        if(sql[s]==32 && s!=y-1);
        else
        {
            if(sql[s] == ',' || s == y-1)
            {
                disp[m][n] = 0;
                m++;
                n=0;
            }
            else
            {
                disp[m][n] = sql[s];
                n++;
            }
        }  /* End of else */
    }  /* End of for */
}  /* End of getdisp() */

/*************************************************************************/
/* Function to check a given display col name for static or time series    */
/* and assign T or S correspondingly to fieldt array.  Also traps          */
/* unsupported  fields.                                                    */
/*************************************************************************/
check_disp(disp)
char disp[];
{
    int x;
    struct fields
    {
        char field[10];
        char fieldt[2];
    };

    static struct fields ipfield[MAX_IP_FIELDS] =
    {
        "COMPNO","S",
        "CO","S",
        "CF","T",
        "NI","T",
        "NS","T",
        "INC","T",
        "ML","S",
        "LI","S",
        "NS","T",
        "ST","T",
        "PER","T",
        "IV","T",
        "ZP","S",
        "EPS","T",
        "PN","T",
        "PC","T",
        "CY","T",
        "SE","S",
        "TA","S",
        "TS","T",
        "AD1","T",
```

```
        "OS","T",
        "CA","T",
        "IT","S",
        "TL","S",
    };

    for(x=0;x<MAX_IP_FIELDS;  x++)
    {
        if(strcmp(disp,  ipfield[x].field)  ==  0)
        {
            return(ipfield[x].fieldt[0]);
        }
    }
    return('0');

}   /* End of check_disp() */

/*****************************************************************************/
/* Function to pick out conditions from SQL query                            */
/*****************************************************************************/
getcond(sql,tcond)
char  sql[400],tcond[MAXCOND][50];
{
    int x, s, m=0, n=0;

    x = searche(sql,"WHERE") + 2;
    /*************************************************************************/
    /* parsing routine to separate each of the conditions and store in arrays  */
    /*************************************************************************/
    for(s=x;  s<strlen(sql);s++)
    {
        if(sql[s]==','||s==strlen(sql)-1||strncmp(sql+s-3," OR
",4)==0||strncmp(sql+s-4,"  AND  ",5)==0)
        {
            tcond[m][n]=32;
            tcond[m][n+1]=0;
            m++;
            n=0;
        }
        else
        {
            tcond[m][n]=sql[s];
            n++;
        }
    }   /* End of for */
}   /* End of get_cond() */

/*****************************************************************************/
/* Function to translate SQL conditions into IP Sharp format                 */
/*****************************************************************************/
trans_cond(tcond)
char  tcond[];
{
    char  temp[50];
    int x, y=1;

    temp[0] = '(';   /* ( at beginning of cond */
    for(x=0;  x<=strlen(tcond);x++)
    {
        if(tcond[x]=='=')
        {
            strcpy(temp+y-1,"  EQ  ");
            y=y+2;
        }
```

```
        else
        {
            if(tcond[x]!=39)   /* " to nothing */
            {
                temp[y]=tcond[x];
                y++;
            }
            if(strncmp(tcond+x-3," OR ",4)==0)      /* Other keywords can be      */
            {                                    /* supported later by just    */
                y=y-4;                           /* adding a similar if        */
                temp[y]=')';                     /* statement and body.        */
                temp[y+1]='\0';
                strcat(temp,tcond+x-3);
                strcpy(tcond,temp);
                return;
            }

            if(strncmp(tcond+x-4," AND ",5) == 0)
            {
                y=y-5;
                temp[y]=')';
                temp[y+1]='\0';
                strcat(temp,tcond+x-4);
                strcpy(tcond,temp);
                return;
            }
        }  /* End of else */
    }  /* End of for */
    strcat(temp,")");
    strcpy(tcond,temp);
}  /* End of trans_cond */


/*****************************************************************************/
/* Function to search for the position of the end of a word within a      */
/* string and return an integer position                                  */
/*****************************************************************************/
searche(sql, prompt)
char prompt[], sql[];
{
    int i, x;

    for(x=0, i=0; x<strlen(sql); x++)
    {
        if (sql[x] == prompt[i])
        {
            if (i == strlen(prompt)-1)
            {
                return(x);
            }
            else
                i++;
        }
    }
    return(0);
}  /* End of searche() */

/*****************************************************************************/
/* Function to search for the position of the beginning of a word within a */
/* string and return an integer position                                  */
/*****************************************************************************/
searchb(sql, prompt)
char prompt[], sql[];
{
```

```
    int i, x;

    for(x=0, i=0; x<strlen(sql); x++)
    {
        if (sql[x] == prompt[i])
        {
            if (i == strlen(prompt)-1)
            {
                return(x-strlen(prompt)+1);
            }
            else
                i++;
        }
    }
    return(0);
}   /* End of searchb */

/*****************************************************************************/
/* Function to create the data file needed by the communications server      */
/*****************************************************************************/
make_file(devpath,lqppath,query, nq)
int nq;
char query[][200], lqppath[], devpath[];
{
    FILE *fp, *fopen();
    char sys[20],filename[100], command[100];
    int x;
    sprintf(filename,"%sipsharp",lqppath);

    sprintf(command,"rm %s",filename);
    system(command);

    if((fp = fopen(filename,"w")) == NULL)
    {
        printf("IP Sharp LQP: Can't comm-server script '%s' for
writing.\n",filename);
        return(NULL);
    }
    fprintf(fp,"cu -s1200 -t 97237165 < %stocomm 2>&1 > %sfromcomm
&\n",devpath,devpath);
    fprintf(fp,"\012\t)\t100\n");
    fprintf(fp,"\007\t)1769739:SLOAN\t100\n");
    fprintf(fp,"\007\t)LOAD 39 MAGIC\t100\n");

    if(strncmp(query[0]+strlen(query[0])-10,"LISTFACTS",9)==0)
    {
        /* For GET COLUMNS */
        fprintf(fp,"a\t%%%sipresults\t100\n",lqppath);
        fprintf(fp,"\007\t%s\t100\n",query[0]);
        fprintf(fp,"a\t%%endcapture\t500\n");
    }
    else
        if(strcmp(query[0],"LIST SERVICES")==0)
        {
            /* For GET TABLES */
            fprintf(fp,"\007\tINFOMAGIC\t100\n");
            fprintf(fp,"a\t%%%sipresults\t100\n",lqppath);
            fprintf(fp,"\007\t%s\t100\n",query[0]);
            fprintf(fp,"a\t%%endcapture\t500\n");
            fprintf(fp,"\007\t)OFF\t500\n");
        }
        else
        {
            fprintf(fp,"\007\tWIDTH 300\t100\n");
```

```
            for(x=0;x<nq;x++)
            {
                if(strncmp(query[x],"PUT",3)==0)
                {
                    fprintf(fp,"\007\tYEARLY DATED 85 TO 89\t100\n");
                    fprintf(fp,"\007\tCOLWIDTH 15\t100\n");
                }
                fprintf(fp,"\007\t%s\t100\n",query[x]);
                fprintf(fp,": \007\tY\t100\n");
                if(strncmp(query[x],"PUT",3)==0)
                    fprintf(fp,"\007\t'IBMPC'TABLE ABOVE\t100\n");
                fprintf(fp,"a\t%%%sipresults\t100\n",lqppath);
                fprintf(fp,"\007\t^M\t100\n");
                fprintf(fp,"a\t%%endcapture\t100\n");
            }
        }
    fprintf(fp,"\007\t)OFF\t500\n");
    fprintf(fp,"\377\t~.\t100\n");
    fclose(fp);

    sprintf(command,"chmod +w %s",filename);
    system(command);
}

/*************************************************************************/
/* Function to read result file and strip beginning and end markers.     */
/* Function also parses data from either table form(time series data)    */
/* or line by line form and consolidates data into one table.            */
/*************************************************************************/
read_file(lqppath,devpath,disp,dispt,dispq, nq, ncol)
char lqppath[], devpath[], disp[][10],dispt[][2],dispq[][3];
int nq, ncol;
{
    FILE   *f1, *f2, *fopen();
    char   start[30], end[30],
           filename[100], command[100],  /* Strings to store full filenames and
commands */
           line[300],                    /* String to read each line of the
result file */
           colname[80],                  /* Temp string for fields in line */
           sdata[MAXCOL][300];           /* String for temporarily storing static
data */

    int    x,y,z1,z2,s=0,   /* generic counter variables */
           lines=0,
           bm=0,     /* beginning marker count */
           em=0,     /* end marker count */
           tsdata=0; /* # of t-s data */

    struct data
    {
        char byte[MAXCHAR];    /* Each token = 15 bytes */
    };
    struct data datatbl[10];  /* Token table = 10 tokens */

    strcpy(start,"*****BEGINNING MARKER*****");
    strcpy(end,"********END MARKER********");

    /* test for existance of file */
    sprintf(filename,"%sipresults",lqppath);
    if((f1 = fopen(filename,"r")) == NULL)
    {
        printf("IP Sharp LQP: Can't open result file '%s' for reading.\n",filename);
        return(NULL);
```

```
    }

    sprintf(filename,"%siptemp",lqppath);

    sprintf(command,"rm  %s",filename);
    system(command);

    if((f2 = fopen(filename,"w")) == NULL)
    {
       printf("IP Sharp LQP: Can't open temporary result file '%s' for
writing.\n",filename);
       return(NULL);
    }
    if(debug==1) printf("IP Sharp LQP: Started reading result file\n");

    /** reading result file for beginning markers to test if   **/
    /** all the data came back                                 **/
    while  (fscanf(f1,"%[^\n]%*c",line)!=EOF)
    {
       if(strncmp(line,start,strlen(start))==0)
          bm++;
       else
          if(strncmp(line+strlen(line)-strlen(end),end,strlen(end))==0)
             em++;
       if(bm>0)                /* Copy to tmp file with */
          fprintf(f2,"%s\n",line);   /* first beg. marker      */
    }

    if(strncmp(disp[0],"TABLE",5)==0)
    {
       if(debug==1) printf("Adding End Marker");
       fprintf(f2,"%s\n",end);
       em++;
    }
    fclose(f1);
    fclose(f2);
    sprintf(command,"chmod  +w  %siptemp",lqppath);
    system(command);
    sprintf(command,"cp  %siptemp  %sipresults",lqppath,lqppath);
    system(command);
    sprintf(command,"chmod  +w  %sipresults",lqppath);
    system(command);

    if(bm != nq || em != nq)
    {
       fprintf(stderr,"READ_FILE: Insufficient results in result file!!\n");
       return(-1);
    }

    if(strcmp(disp[0],"NO")==0  &&  strcmp(disp[1],"MNEM")==0  &&
strcmp(disp[2],"DESC")==0)
    {
       bm = 0;
       em = 0;
       sprintf(filename,"%sipresults",lqppath);
       if((f1 = fopen(filename,"r")) == NULL)
       {
          printf("IP Sharp LQP: Can't open result file '%s' for
reading.\n",filename);
          return(NULL);
       }
       if(debug==1) printf("IP Sharp LQP - Reading ipresults\n");

       sprintf(filename,"%sconnect1.tmp",lqppath);
```

```
        sprintf(command,"rm  %s",filename);
        system(command);
        if((f2 = fopen(filename,"w")) == NULL)
        {
           printf("IP Sharp LQP: Can't open temporary result file '%s' for
writing.\n",filename);
           return(NULL);
        }
        fprintf(f2,"%s\n",start);
        fclose(f2);

        while  (fscanf(f1,"%[^\n]%*c",line)!=EOF)
        {
           if(strncmp(line,start,strlen(start))==0)
           {
              lines=0;
              bm++;
           }
           else
              if(strncmp(line+strlen(line)-strlen(end),end,strlen(end))==0)
              {
                 em++;
              }
              else
                 if(strlen(line)>19)
                 {
                    /* find corresponding columns     */
                    f2=fopen(filename,"a");
                    strncpy(colname,line+3,4);
                    colname[4]=0;
                    fprintf(f2,"%s|",colname);
                    strncpy(colname,line+8,8);
                    colname[8]=0;
                    fprintf(f2,"%s|",colname);
                    sscanf(line+16,"%[^\r]%*c",colname);
                    fprintf(f2,"%s|\n",colname);
                    fclose(f2);
                 }
        }  /* End of while */
     }  /* End of if */
     else
        if(strcmp(disp[0],"TABLE")==0  &&  strcmp(disp[1],"DESC")==0)
        {
           bm = 0;
           em = 0;
           sprintf(filename,"%sipresults",lqppath);
           if((f1 = fopen(filename,"r")) == NULL)
           {
              printf("IP Sharp LQP: Can't open result file '%s' for
reading.\n",filename);
              return(NULL);
           }

           if(debug==1) printf("IP Sharp LQP - Reading ipresults\n");

           sprintf(filename,"%sconnect1.tmp",lqppath);
           sprintf(command,"rm  %s",filename);
           system(command);

           f2=fopen(filename,"a");
           fprintf(f2,"%s\n",start);
           fclose(f2);

           while  (fscanf(f1,"%[^\n]%*c",line)!=EOF)
```

```
        {
            for(s=0;s<strlen(line);s++)
            {
                if(line[s]=='-')
                break;
            }

            if(strncmp(line,start,strlen(start))==0)
            {
                lines=0;
                bm++;
            }
            else
                if(strncmp(line+strlen(line)-strlen(end),end,strlen(end))==0)
                {
                    em++;
                }
                else
                    if(s<strlen(line) && s > 3)
                    {
                        /* find corresponding disp columns      */
                        f2=fopen(filename,"a");
                        strncpy(colname,line,s-1);
                        colname[s-1]=0;
                        fprintf(f2,"%s|",colname);
                        sscanf(line+s+1,"%[^\r]%*c",colname);
                        fprintf(f2,"%s|\n",colname);
                        fclose(f2);
                    }
        }  /* End of while */
    }  /* End of if */
    else
    {
        /* Reading result file to get the data while keeping track of  */
        /* beginning and end markers */
        bm = 0;
        em = 0;

        sprintf(filename,"%sipresults",lqppath);
        if((f1 = fopen(filename,"r")) == NULL)
        {
            printf("IP Sharp LQP: Can't open result file '%s' for
reading.\n",filename);
            return(NULL);
        }

        if(debug==1) printf("IP Sharp LQP - Reading ipresults\n");

        sprintf(filename,"%sconnect1.tmp",lqppath);
        sprintf(command,"rm %s",filename);
        system(command);

        f2=fopen(filename,"a");
        fprintf(f2,"%s\n",start);
        fclose(f2);

        while (fscanf(f1,"%[^\n]%*c",line)!=EOF)
        {
            if(strncmp(line,start,strlen(start))==0)
            {
                lines=0;
                bm++;
            }
            else
```

```
                if(strncmp(line+strlen(line)-strlen(end),end,strlen(end))==0)
                {
                    em++;
                }
                else
                {
                    /* find corresponding disp columns      */
                    for(x=0;disp[x][0]!=0;x++)
                    {
                        if(dispq[x][0]-48==bm) /* find disp col for this query */
                        {
                            if(dispt[x][0]=='S') /* check if static data */
                            {
                                /* Check for static data less than 2 lines */
                                if(++lines > 1) /* make sure lines=1    */
                                {
                                    printf("IP Sharp - Too many data lines for:
%s\n",disp[x]);
                                    return(-1);
                                }
                                z1=0;z2=0;
                                do
                                {
                                    if  (line[z1]!=13)
                                    {
                                        sdata[bm-1][z2]=line[z1];
                                        z2++;
                                    }
                                    z1++;
                                } while(sdata[bm-1][z2-1]!=0) ;/* assign to static
data[n]*/
                            }
                            if(dispt[x][0]=='T') /* check if time-series data */
                            {
                                if(count(line,32)!=strlen(line)-1  &&
strncmp(line+strlen(line)-6,"col",3)!=0)
                                {
                                    tsdata=dataize(line,datatbl,ncol);   /* parse
lines and assign to datatbl[n] str*/

                                    /* write data to a file */
                                    for(y=0;y<bm-1;y++)
                                    {
                                        f2=fopen(filename,"a");
                                        fprintf(f2,"%s|",sdata[y]);
                                        fclose(f2);
                                    }

                                    for(y=0;y<tsdata;y++)
                                    {

                                        f2=fopen(filename,"a");
                                        fprintf(f2,"%s|",datatbl[y].byte);
                                        fclose(f2);
                                    }
                                    f2=fopen(filename,"a");
                                    fprintf(f2,"\n");
                                    fclose(f2);
                                    break;
                                }  /* End of if */
                            }  /* End of if */
                        }  /* End of if */
                    } /* end of for() */
                }  /* End of else */
```

```
            }  /* End of while() */
            if(tsdata==0)
              {
                   f2=fopen(filename,"a");
               for(y=0;y<bm;y++)
                 {
                   fprintf(f2,"%s|",sdata[y]);
                 }
               fprintf(f2,"\n");
                   fclose(f2);
            }
        }  /* End of else */
        f2=fopen(filename,"a");
        fprintf(f2,"%s\n",end);
        fclose(f2);
        fclose(f1);

        return(1);
}  /* End of read_file() */


/*************************************************************************/
/* Function that parses time series data by line and stores data in a table*/
/*************************************************************************/
dataize (line, datatbl)
char line[MAXCHAR];
struct data
{
    char byte[MAXCHAR];    /* Each token = 15 bytes */
};
struct data datatbl[10];   /* Token table = 10 tokens */

{
    int x,i=0,j=0;
    for(x=0;x<strlen(line);x++)
    {
        if(line[x]!=' ')   /* skip spaces */
        {
            datatbl[i].byte[j]=line[x];
            j++;
        }
        if((x+1)%MAXCHAR==0)    /* max possible length for data */
        {
            datatbl[i].byte[j]=0;
            i++;
            j=0;
        }
    }
    return(i);
}  /* End of dataize */

/*****************************************************************/
/* Function to count the occurance of the letter within the line */
/*****************************************************************/
count(line,letter)
char line[];
int letter;
{
   int x=0, y=0;
   while(line[x]!=0)
     {
        if(line[x]==letter)
    y++;
        x++;
```

```
    }
  return(y);
}  /* End of count */
```

ipsharp.lsp

```
; ****************************
; **  FILE: ipsharp.lsp  **
; ****************************
; By Alec R. Champlin (April, 1988)
; As part of MIT Undergrad. Thesis
;
;
; Modified by:
; Francis C.K. Gan (May, 1989)
; As part of MIT Undergrad. Thesis
;
; For use as the I.P. Sharp LQP
;
;
;;; **MODIFICATIONS*
;;; 06/10/88 - tk
;;; changed Comm server directory in object

; I.P. Sharp LQP Manager and front end for C program
;
; The IPSHARP object can respond to the following messages:
;      =>     :self-info
;      =>     :get-tables
;      =>     :get-columns <table-name>
;      =>     :get-data <cis/tk-single-query>

(defun display-IPSHARP-self-info  ()
  (if (y-or-n-p "Do you want to see a description of the INFORMIX RDBMS?")
      (progn (format t "~%
                  I. P. S H A R P / D I S C L O S U R E
                ------------------------------------------~%
The I.P. Sharp is a commercial database located in Toronto, CANADA.  The
I.P.
Sharp service consists of access to various databases.  The data is
retreived
using its own query language which is similar in structure to SQL though
less
powerful.  However, the I.P. Sharp database is NOT relational.  The
Disclosure
database contains corporate and earnings information on most companies in
the
United States.~%~%")
      (format t "
                     Press any key to continue.~%")
      (read-char))))

(defun get-IPSHARP-tables ()
  (let ((result (connect (get-current-object) "INFO TABLES"))
      (comdir (get-self 'comm-server-directory)))
    (lqp-print 'terse "Filtering out irrelevant communications
messages...")
    (system (format nil "~A/filter ~A" comdir result))
    (lqp-print 'terse "Done.~%")
    (lqp-print 'verbose "DBMS result file....~%")
    (lqp-print-file 'verbose result)
    (lqp-print-file 'normal result)
```

```
    (read-standard-table result comdir)))

(defun get-IPSHARP-columns (table)
  (let ((result (connect (get-current-object)
                    (format nil
                          "INFO COLUMNS FOR ~A"
                          (parse-SQL-tname table))))
      (comdir (get-self 'comm-server-directory)))
    (lqp-print 'terse "Filtering out irrelevant communications
messages...")
    (system (format nil "~A/filter ~A" comdir result))
    (lqp-print 'terse "Done.~%")
    (lqp-print 'verbose "DBMS result file....~%")
    (lqp-print-file 'verbose result)
    (lqp-print-file 'normal result)
    (read-standard-table result comdir)))

; THE "EFFICIENT" ARGUMENT TO FUNC. CONNECT TELLS IT TO USE A SCRIPT THAT
; MAKES INFORMIX DO THE "STANDARDIZING" OF TABLES FOR ME.
(defun get-IPSHARP-data (abs_local_query)
  (multiple-value-bind (SQL columns) (form-SQL abs_local_query)
      (lqp-print 'normal "SQL query to be sent to DBMS....~%~A~%" SQL)
      (lqp-print 'verbose "Columns reported by FORM-SQL...~%~A~%" columns)
      (let ((result (connect (get-current-object) SQL 'EFFICIENT))
            (comdir (get-self 'comm-server-directory)))
        (lqp-print 'terse "Filtering out irrelevant communications ~
                           messages...")
        (system (format nil "~A/filter ~A" comdir result))
        (lqp-print 'terse "Done.~%")
        (lqp-print 'normal "Result file after conversion to 'standard' ~
                            form...~%")
        (lqp-print-file 'normal result)
        (cons columns (read-standard-table result comdir)))))


;-------------------------------------------------------------------------;
;            DEFINITION OF LQP OBJECT CLASS IPSHARP                       ;
;-------------------------------------------------------------------------;

; (in curr-lqp file)
;-------------------------------------------------------------------------;
;        END OF DEFINITION OF LQP IPSHARP   (IPSHARP.LSP)                 ;
;-------------------------------------------------------------------------;
```

Appendix B: Modified Existing Code

3b2fetch.c (2AFETCHE and 2CFETCHE)

2AFETCHE and 2CFETCHE that used to be shell script are now data files that are used by commserv1. 2AFETCHE and 2CFETCHE are now created by 2AFETCHE.c and 2CFETCHE.c respectively and a copy of the file (they are both identical) now known as .c (executable is 3BFETCH).

```
#include<stdio.h>
#include<string.h>
main(argc,argv)
    int argc;
    char *argv[];
{
    FILE *fp, *fopen();
    int x;
    char filename[100],/*actually the filename*/
          command[100],
          lqproot[100],
          temppath[100],
          devpath[80];

/* Will be read from .profile or something like that for the final version */
    strcpy(temppath,"/usr/tmp/");
    strcpy(lqproot,"/usr/cistk/demo/v2/lqp/");
    strcpy(devpath,"/usr/cistk/demo/v2/lqp/dev/");

    sprintf(filename,"/usr/cistk/demo/v2/lqp/dev/%s",argv[1]);
    sprintf(command,"rm %s",filename);
    system(command);
    fp = fopen(filename,"w");
    fprintf(fp,"telnet %s < %stocomm > %sfromcomm &\n", argv[1], devpath, devpath);
    fprintf(fp,"login:\t%s\t100\n",argv[2]);
    fprintf(fp,"Password:\t%s\t100\n",argv[3]);
    fprintf(fp,"is:\t%s\t100\n",argv[4]);
    fprintf(fp,"$\tcd %s\t100\n",argv[5]);
    fprintf(fp,"$\techo \"UNLOAD TO %sresults.tmp %s\" | %s\t100\n", temppath,
argv[7], argv[6]);
    fprintf(fp,"$\t%%%sconnect1.tmp\t100\n",lqproot);
    fprintf(fp,"$\tcat %sresults.tmp\t100\n",temppath);
    fprintf(fp,"$\t%%endcapture\t100\n");
    fprintf(fp,"$\tlogout\t100\n");
    fclose(fp);
    sprintf(command,"chmod +w %s", filename);
          system(command);
    sprintf(command,"%scommserv1 %s", devpath, argv[1]);
    system(command);
          system("rm %s*comm", devpath);
    system("cp %sconnect1.tmp %sconnect2.tmp", lqproot, lqproot);

}
```

strip-rt.c

strip-rt.c has been modified to work better and also to accomodate the new beginning and end markers in the result file. The following is the new version;

```
/*************************************************************************/
/**                                                                     **/
/**    FILE: STRIP-RT.C      By Alec R. Champlin (April, 1988)          **/
/**   ==================     As part of MIT Undergrad. Thesis           **/
/**                                                                     **/
/** THIS ROUTINE FILTERS OUT EXTRANEOUS STUFF FROM THE ORACLE RESULT    **/
/**                                                                     **/
/*************************************************************************/

#include <stdio.h>
#define MAXLINE 100

index(s,t)
char s[], t[];
{
  int i, j, k;

  for (i=0; s[i] != '\0'; i++)
    {
       for (j=i, k=0; t[k] !='\0' && s[j]==t[k]; j++, k++)
   ;
      if (t[k] == '\0')
   return(i);
    }
  return(-1);
}

main (argc, argv)
     int  argc;
     char *argv[];
{
  int           i1, i2, x;
  FILE           *in_file, *tmp_file, *out_file;
  char           l[MAXLINE],tmp[30];
  static char   start[] = { "Doesn't make any difference" };
  static char   blanks[] = { "\n" };
  static char   end[] = {"selected."};
  static char   cols[]= {"-\n"};
  if ( argc != 2 )
    {
       printf ("STRIP-RT -- Expecting one argument, <FILENAME>.\n");
      exit (1);
    }
  else if ( (in_file = fopen (argv[1], "r+")) == NULL )
    {
       printf ("STRIP-RT -- Couldn't open file \"%s\".\n", argv[1]);
      exit (2);
    }
  else if ( (tmp_file = tmpfile ()) == NULL)
    {
       printf ("STRIP-RT -- Couldn't open temporary file.\n");
      exit (3);
    }

  i2 = -1;
  while ( i2 == -1 )
    {

       if ((fgets(l,MAXLINE,in_file)) == NULL)
```

```
    {
      printf ("STRIP-RT -- End marker not found.\n");
      exit (4);
    }

        if(strncmp(l+strlen(l)-1-strlen(end),end,strlen(end))==0)
       i2=0;
        printf("%s\n%s\n%d\n",l+strlen(l)-1-strlen(cols),cols,strlen(cols));
       if ((strcmp(l,blanks)) != 0 && strncmp(l+strlen(l)-
strlen(cols),cols,strlen(cols))!=0)
    {
        fputs(l, tmp_file);
    }

     }
   fputs ('\n',tmp_file);
   if ( (out_file = freopen (argv[1], "w", in_file)) == NULL)
     {
       printf ("STRIP-RT -- 'FREOPEN' Error.\n");
      exit (5);
     }

   rewind (tmp_file);
   i2 = -1;
   while ( i2 == -1 )
     {
       fgets(l, MAXLINE, tmp_file);
      i2 = index(l,end);
      if (i2 == -1)
    {
      if ( (fputs (l, out_file)) == EOF )
        {
          printf ("STRIP-RT -- 'FPUT' Error.\n");
         exit (6);
        }
    }
     }
}
```

filter.c

filter.c has been modified to accomodate the new format for beginning and end markers.

```
/*************************************************************************/
/**                                                                     **/
/**    File: FILTER.C      By Alec R. Champlin (April, 1988)            **/
/**   ===============     As part of MIT Undergrad. Thesis              **/
/**                                                                     **/
/**                        Modified by:                                 **/
/**                         Francis C.K. Gan (April, 1989)              **/
/**                                                                     **/
/**                                                                     **/
/**     THIS ROUTINE WILL FILTER OUT EVERYTHING BEFORE A BEGINNING      **/
/**          MARKER AND EVERYTHING AFTER AN ENDING MARKER               **/
/**                                                                     **/
/*************************************************************************/

#include <stdio.h>
#define MAXTMP 300
main (argc, argv)
     int  argc;
     char *argv[];
{
  int           c=0, i1, i2, foundbeg=0;
  FILE          *in_file, *tmp_file, *out_file;
  char          tmp[MAXTMP];
  static char   start[] = { "*****BEGINNING MARKER*****" };
  static char   end[] = { "********END MARKER********" };

  if ( argc != 2 )
    {
       printf ("FILTER -- Expecting one argument, <FILENAME>.\n");
      exit (1);
    }
  else if ( (in_file = fopen (argv[1], "r+")) == NULL )
    {
       printf ("FILTER -- Couldn't open file \"%s\".\n", argv[1]);
      exit (2);
    }
  else if ( (tmp_file = tmpfile ()) == NULL)
    {
       printf ("FILTER -- Couldn't open temporary file.\n");
      exit (3);
    }
   i2 = strlen (start);
   for ( i1 = 0; foundbeg!=1 || c!='\n'; )
    {
      if ( (c = getc (in_file)) == EOF )
   {
     rewind (in_file);
     break;
    }
       if (c == start[i1])
      {
    ++i1;
        if(i1==i2-1)
       {
          foundbeg=1;
    }
      }
      else
    i1 = 0;
    }
   i2 = strlen (end);
```

```
for(i1=0;i1<MAXTMP;)
{
   if ( (c = getc (in_file)) == EOF)
   {
       for (i1 = 0; tmp[i1] != '\0' ; ++ i1)
           fputc (tmp[i1], tmp_file);
        printf ("FILTER -- End marker not found.\n");
       break;
   }
   tmp[i1] = c;
   tmp[++i1] = '\0';
    if(strncmp(tmp+i1-strlen(end),end,strlen(end))==0)
   {
      fputc ('\n',tmp_file);
     break;
   }
   else
       if(c==10)
       {
          for (i1 = 0; tmp[i1] != '\0' ; ++ i1)
         {
       if(tmp[i1]==13)
         i1++;
       fputc (tmp[i1], tmp_file);
    }
         i1=0;
       }
}
if ( (out_file = freopen (argv[1], "w", in_file)) == NULL)
  {
     printf ("FILTER -- 'FREOPEN' Error.\n");
    exit (5);
  }
rewind (tmp_file);
while ( (c = getc (tmp_file)) != EOF )
  {
     if ( (fputc (c, out_file)) == EOF )
 {
    printf ("FILTER -- 'FPUT' Error.\n");
    exit (6);
 }
  }
}
```

Connect.lsp

Connect.lsp has been modified to issue arguments to the new 3BFETCH in the correct order.

```
(* Note, this file has always been and still is tailored for the informix machines only
that need the h19 terminal type)
; ************************
; **  FILE: CONNECT.LSP  **
; ************************
; By Alec R. Champlin (April, 1988)
; As part of Undergrad. Thesis

;-----------------------------------------------------------------;
;      ROUTINES FOR CONTROLLING THE DBMS COMMUNICATION SCRIPTS     ;
;-----------------------------------------------------------------;

(defun connect (DBMS-obj SQL &optional (use-eff? nil))
  (let* ((local    (get-object DBMS-obj 'local-DBMS?))
         (comdir   (get-object DBMS-obj 'comm-server-directory))
         (dbdir    (get-object DBMS-obj 'database-directory))
         (db       (get-object DBMS-obj 'database))
         (dbtype   (get-object DBMS-obj 'type-of-DBMS))
         (machine  (get-object DBMS-obj 'machine-name))
         (account  (if (not local)
                       (get-object DBMS-obj 'account)
                       "IRRELEVANT"))
         (passwrd  (if (not local)
                       (get-object DBMS-obj 'password)
                       "IRRELEVANT"))
         (script   (if use-eff?
                       (get-object DBMS-obj 'efficient-comm-script)
                       (get-object DBMS-obj 'communications-script)))
         (tmpfile1 (si:string-concatenate comdir "/connect1.tmp"))
         (tmpfile2 (si:string-concatenate comdir "/connect2.tmp"))
         invoker)
    (setq script (format nil "~A/~A" comdir script))
    (cond ((or (equal dbtype 'INFORMIX) (equal dbtype "Informix")
               (equal dbtype "INFORMIX") (equal dbtype "informix"))
           (setq invoker (format nil "isql ~A -" db)))
          ((or (equal dbtype 'ORACLE) (equal dbtype "Oracle")
               (equal dbtype "ORACLE") (equal dbtype "oracle")
               (equal dbtype "SQL/RT") (equal dbtype "sql/rt"))
           (setq invoker "sqlcmd"))
          (t (lqp-print 'terse
                        "CONNECT -- Database type ~A unrecognized. ~
                         Update CONNECT.LSP with new dbtype.~%" dbtype)
             return))
    (lqp-print 'terse "Connecting to ~A on machine ~A..." db machine)
    (if (not local)
        (system (unix-format "~A ~A ~A ~A h19 ~A ~A ~A"
                             script machine account passwrd dbdir invoker SQL))
        (system (unix-format "~A ~A ~A ~A ~A ~A 1> ~A 2> ~A"
                             script account passwrd dbdir invoker SQL
                             tmpfile1 tmpfile2)))
    (lqp-print 'terse "Done.~%")
    (values tmpfile1 tmpfile2)))

(defun unix-format (str &rest args)
  (setq args (mapcar #'(lambda (x) (format nil "~C~A~C" #\" x #\"))
                     args))
  (apply #'format (cons nil (cons str args))))

;-----------------------------------------------------------------;
```

```
;       END OF COMMUNICATIONS SCRIPT CONTROL ROUTINES (CONNECT.LSP)       ;
;-------------------------------------------------------------------------;
```

Appendix C: How to use the communications server

The first stage implementation of the communications server should be used by LQP that do not require dynamic or interaction with the foreign host. In this case, the intelligence of the communications server is dependent upon the script generated by the LQP for the communications server. On the other hand, the stage two implementation allows an LQP to interact in real time with the foreign host. This time, the LQp has direct control over the intelligence of the communications server. This appendix will discuss how to use the Stage One implementation.

The format for the data file needed by the Stage One version of the communications server is as follows:

```
commandline
prompt1, response1, timeout1
prompt2, response2, timeout2
          .
          .
          .
promptN, responseN, timeoutN
```

A file like this can be created in many ways. However, it has to be dynamically created for each session. The key for creating such a file is knowing the login and query procedures for any given foreign host. Let's use the mit2c script as an example.

```
telnet mit2c < tocomm > fromcomm &
login:  demo     100
Password:        cis/tk  100
is:      h19     100
$        cd /usr/pagetm/cis      100
$         UNLOAD TO /usr/tmp/results.tmp SELECT CO, COMPNO FROM
GENINFO WHERE COMPNO = 2530
$        %/usr/cistk/demo/lqp/v1/connect1.tmp    100
$        cat /usr/tmp/results.tmp        100
$        %endcapture     100
$        logout  100
```

With the exception of the first line in the script, the rest of the file is a three column file delimited with tabs (\t) for each column and each line is separated with a linefeed(\n). The first column should contain the expected prompt, the second column the corresponding response and the third column the estimated timeout. The first line of the file is the command line that should go to the system to initiate the communications. Currently, only two communications that are know are supported: cu and telnet (although any communications software that is compatible with pipes should work). If you are using telnet, the only thing you have to have different in the command line is the name of the host you need to connect to. The name of the host has to be acceptable to telnet. One way to ensure this is to try "telnet <hostname>" from the UNIX prompt. If you need cu to connect to your foreign host, the command line is similar with the exception that you will probably need to combine the stdout and stderr. This can be done as follows:

```
cu -s1200 -h -t 9777777 < tocomm 2>&1 > fromcomm &
```

The -h and -t options are specific for the I.P. Sharp database. You should find out what communications protocols your foreign host needs and use the corresponding options with cu. In such a case, the command line will will have different '-' options and a different phone number (remember to add the 9 to the beginning of the phone number if using an institute phone line for the modem).

The three columns of data is easy if you know the foreign host well. First, log on to the foreign system in real time and capture the session into a file. Perform a sample query and whatever you would like the system to normally do while retrieving data. When you are done, log out and look at a printout of the session. On a separate peice of paper divided into three columns:

i) put a section of text which is uniquely identifiable up to the point where a response is sent into the first column (watch out for escape or control characters that may not necessarily show on the screen). Sometimes, some data may be received through stderr as opposed to stdout. This is reconciled or combined using the method shown above for cu but not for telnet because no need has arised yet. Unless it is proved necessarily for a specific case, it is not recommended to combine the stderr and stdout from telnet because telnet tends to have more erratic timing than cu and intercepting the stderr messages as well may cause the communications server to miss the expected prompt.

ii) in the second column write the response you would normally type in when you see the prompt in the first column

iii) in the last column, write a reasonable timeout period in seconds before the communications server should timeout if it does not see the next prompt specified in the file. The timeout should take into account that some queries could potentially be very long. In such a case, have a longer timeout. It is safer to have a longer timeout than a shorter one

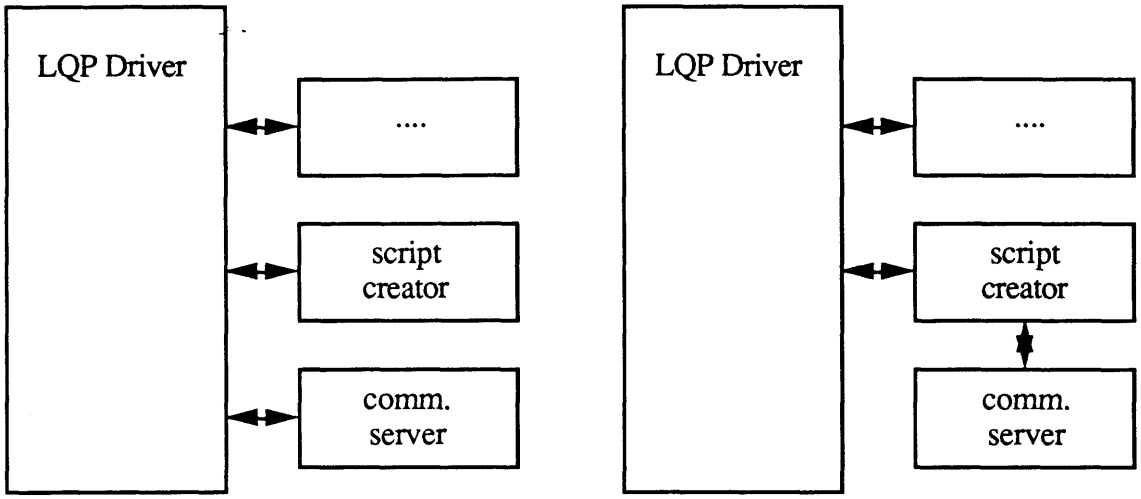
When you are done, you now have to include the capture to file and end capture toggles. The capture toggle should be inserted before the line that contains the response to tell the foreign host to start displaying the results (this does not necessarily have to be the query iteself because on some systems like I.P. Sharp, the system prompts after it has found some data. The format for the capture toggle is the filename you would like to write the data to preceeded by a '%'. The only exception that you cannot use for a filename is "endcapture" because that is used as the keyword for the endcapture toggle. The end capture toggle should be inserted after the line that contains the response to tell the foreign host to start

displaying the results. The format for the capture and endcapture toggle lines is the same as that for the other normal lines except for the fact that in the prompt column, any dummy stuff can be put there. The timeout is also irrelevent. In the case above, the capture toggle is as follows:

```
$       %/usr/cistk/demo/lqp/v1/connect1.tmp     100
```

Once this table is complete, write a program in your computer mother tongue or whatever language you are using to create this file. See the sample program in Appendix A that creates the script file for MIT2C. Depending on what your LQP supports, you should use variables for responses like the loginind, password and the query itself. Sometimes, the order of your prompts and responses may have to be dynamic based on your query. In such a case, you should make sure that lines in the file created by your program are logically arranged. Try to ensure that the foreign host session and the communications program you are using terminates gracefully.

In your LQP itself, after you have called/run your communications server script creator (see Figure C1), you can invoke the communications server by issuing a system command to the effect of 'commserv <script>' where the name of the script should be the name of the telnet hostname if you are using telnet.



Correct LQP call to communications server

Existing LQP call to communications server for some LQPs

Figure C1: LQP call to communication server.

Appendix D: Sample Run of CIS/TK using the I.P. Sharp LQP

The following sample runs are done from within IBCL with the debug flag in the communications server and the I.P. Sharp LQP set to on.

```
(print lqp-ipsharp1)

(SEND-MESSAGE 'DISCLOSUREDB :GET-DATA
     '(DISCLOSURE (CO COMPNO NI CF) (= COMPNO "3130")))
(SEND-MESSAGE 'DISCLOSUREDB :GET-DATA
     '(DISCLOSURE (CO COMPNO NI CF) (= COMPNO "3130")))

>(eval lqp-ip[sharp       sharp1)
   1> (SEND-MESSAGE DISCLOSUREDB :GET-DATA
          (DISCLOSURE (CO COMPNO NI CF) (= COMPNO "3130")))
      2> (SYSTEM "\"/usr/cistk/demo/v2/lqp/ipsharp/qt\" \"foreign\" \"SLOAN:...\"
\"...\" h19 \"/usr/bin\" \"isql adisclosure -\" \"SELECT CO, COMPNO, NI, CF FROM
DISCLOSURE WHERE COMPNO = '3130'\"")
IPSharp LQP - Query Received: SELECT CO, COMPNO, NI, CF FROM DISCLOSURE WHERE
COMPNO = '3130'
CO S
COMPNO S
NI T
CF T
(COMPNO EQ 3130 )
IPSharp LQP - Queries generated:
DISPLAY '(COMPNO EQ 3130 )' ADISCLOSURE 'CO'
DISPLAY '(COMPNO EQ 3130 )' ADISCLOSURE 'COMPNO'
PUT '(COMPNO EQ 3130 )' ADISCLOSURE 'NI,CF'
Finished reading file
Connected


)

  GHBGSBHSGBSBGBSGBM4M4M4M4MNMNMNMNM58585858HEHEHEHEH)1769739:SLOAN

   9871) 1989-05-15 17:12:04 IPSA

SHARP APL SERVICE

       )LOAD 39 MAGIC

SAVED   1989-05-15 01:44:37

       WIDTH 300

        DISPLAY '(COMPNO EQ 3130 )' ADISCLOSURE 'CO'

1 FOUND. PROCEED? (Y/N): Y
```

```
FORD MOTOR CO
       DISPLAY '(COMPNO EQ 3130 )' ADISCLOSURE 'COMPNO'

1 FOUND. PROCEED? (Y/N): Y



3130
       YEARLY DATED 85 TO 89

       COLWIDTH 15

       PUT '(COMPNO EQ 3130 )' ADISCLOSURE 'NI,CF'

1 FOUND. PROCEED? (Y/N): Y

       'IBMPC'TABLE ABOVE

       2,515,400      19,851,231
       3,285,100      19,861,231
       4,625,200      19,871,231
       5,300,200      19,881,231

           col 1          col 2
)OFF

   9871   1989-05-15 17:12:53 IPS
CONNECTED      00:00:48  TO DATE      03:49:22
CPU UNITS       137.603  TO DATE     10084.249
KILOCHARS         0.716  TO DATE       337.085


Lost Carrier


Disconnected
```

```
IP Sharp LQP: Started reading result file
IP Sharp LQP - Reading ipresults
    <2 (SYSTEM 0)
     2> (SYSTEM "/usr/cistk/demo/v2/lqp/ipsharp/filter
/usr/cistk/demo/v2/lqp/ipsharp/connect1.tmp")
    <2 (SYSTEM 16777215)
     2> (SYSTEM "/usr/cistk/demo/v2/lqp/ipsharp/preREAD
/usr/cistk/demo/v2/lqp/ipsharp/connect1.tmp")
    <2 (SYSTEM 16777215)
  <1 (SEND-MESSAGE
        (("CO" "COMPNO" "NI" "CF")
          ("FORD MOTOR CO" "3130" "3,285,100" "19,861,231")
          ("FORD MOTOR CO" "3130" "4,625,200" "19,871,231")
          ("FORD MOTOR CO" "3130" "5,300,200" "19,881,231")))
(("CO" "COMPNO" "NI" "CF")
 ("FORD MOTOR CO" "3130" "3,285,100" "19,861,231")
 ("FORD MOTOR CO" "3130" "4,625,200" "19,871,231")
 ("FORD MOTOR CO" "3130" "5,300,200" "19,881,231"))

>(printf  lqp-ipsharp3)

(SEND-MESSAGE 'DISCLOSUREDB :GET-COLUMNS '("disclosure"))
(SEND-MESSAGE 'DISCLOSUREDB :GET-COLUMNS '("disclosure"))

>(eval lqp-ipsharp3)
  1> (SEND-MESSAGE DISCLOSUREDB :GET-COLUMNS ("disclosure"))
    2> (SYSTEM "\"/usr/cistk/demo/v2/lqp/ipsharp/qt\" \"foreign\" \"SLOAN:...\"
\"...\" h19 \"/usr/bin\" \"isql adisclosure -\" \"INFO COLUMNS FOR disclosure\"")
IPSharp LQP - Query Received: INFO COLUMNS FOR disclosure
Finished reading file
Connected



)

 GHBGSBHSGBSBGBSGBM4M4M4M4MNMNMNMNM58585858HEHEHEHEH)1769739:SLOAN

  9898) 1989-05-15 17:14:07 IPSA

SHARP APL SERVICE

      )LOAD 39 MAGIC

SAVED  1989-05-15 01:44:37

       ADISCLOSURE 'LISTFACTS'

   NO.  MNEMONIC  DESCRIPTION


TIMESERIES FACTS:

     1  CH      CASH

     2  MS       MARKETABLE SECURITIES
```

```
  3  RE      RECEIVABLES
  4  IV      INVENTORIES
  5  RM      RAW MATERIALS
  6  WP      WORK IN PROGRESS
  7  FG      FINISHED GOODS
  8  NR      NOTES RECEIVABLE
  9  OC      OTHER CURRENT ASSETS
 10  CA      CURRENT ASSETS
 11  PR      PROPERTY; PLANT + EQUIPMENT
 12  DP      ACCUMULATED DEPRECIATION
 13  PN      NET PROPERTY; PLANT + EQUIPMENT
 14  IA      INVESTMENT + ADVANCES TO SUBS
 15  NC      OTHER NON-CURRENT ASSETS
 16  DC      DEFERRED CHARGES
 17  IS      INTANGIBLES
 18  DS      DEPOSITS + OTHER ASSETS
 19  TA      TOTAL ASSETS
101  NP      NOTES PAYABLE
102  AP      ACCOUNTS PAYABLE
103  CD      CURRENT LONG TERM DEBT
104  CL      CURRENT PORTION OF CAP LEASES
105  AE      ACCRUED EXPENSES
106  IC      INCOME TAXES
107  RL      OTHER CURRENT LIABILITIES
108  LI      TOTAL CURRENT LIABILITIES
109  MG      MORTGAGES
110  DF      DEFERRED CHARGES/INCOME
111  CV      CONVERTIBLE DEBT
112  LD      LONG TERM DEBT
113  NL      NON-CURRENT CAPITAL LEASES
114  LL      OTHER LONG TERM LIABILITIES
115  TL      TOTAL LIABILITIES
```

```
116  ML      MINORITY INTEREST (LIABILITIES)
117  PS      PREFERRED STOCK
118  SN      COMMON STOCK NET
119  SR      CAPITAL SURPLUS
120  RT      RETAINED EARNINGS
121  TR      TREASURY STOCK
122  OL      OTHER LIABILITIES
123  SE      SHAREHOLDER'S EQUITY
124  LW      TOTAL LIABILITIES + NET WORTH
201  NS      NET SALES (INCOME STATEMENT)
202  CG      COST OF GOODS
203  GP      GROSS PROFIT
204  RD      R + D EXPENDITURES
205  SG      SELL; GENERAL + ADMIN EXPENSES
206  ID      INCOME BEFORE DEPREC + AMORT
207  DA      DEPRECIATION + AMORTIZATION
208  NO      NON-OPERATING INCOME
209  IE      INTEREST EXPENSE
210  IB      INCOME BEFORE TAX
211  PT      PROVISION FOR INCOME TAXES
212  MI      MINORITY INTEREST
213  IG      INVESTMENT GAINS/LOSSES
214  OI      OTHER INCOME
215  IX      NET INCOME BEFORE EXTRA ITEMS
216  EI      EXTRA ITEMS + DISCONTINUED OPS
217  NI      NET INCOME (INCOME STATEMENT)
218  OS      OUTSTANDING SHARES
301  CF      FINANCIAL REPORTING DATE (YYYYMMDD)
302  IBEX    INCOME (LOSS) BEFORE EXTRAORDINARY ITEMS
303  DEP     DEPRECIATION AND DEPLETION
304  DEFT    DEFERRED INCOME TAX
305  MININ   MINORITY INTEREST/EQUITY IN SUBSIDIARIES AND AFFI
```

306	OFO	OTHER FUNDS FROM OPERATIONS
307	TFO	TOTAL FUNDS PROVIDED BY OPERATIONS
308	FUEX	FUNDS USED FOR EXTRAORDINARY ITEMS
309	SPPE	SALES OF PROPERTY; PLANT AND EQUIPMENT
310	ILTD	ISSUANCE OF LONG TERM DEBT
311	SOS	SALE OF STOCK
312	OSF	OTHER SOURCES OF FUNDS
313	TSF	TOTAL SOURCES OF FUNDS
314	DIV	DIVIDENDS
315	CEXP	CAPITAL EXPENDITURES
316	ININV	INCREASE IN INVESTMENT
317	DECLTD	DECREASE IN LONG TERM DEBT
318	PURCS	PURCHASE OF STOCK
319	ACQNS	ACQUISITIONS
320	OUF	OTHER USES OF FUNDS
321	TUF	TOTAL USES OF FUNDS
322	INCWC	INCREASE (DECREASE) IN WORKING CAPITAL
401	QKR	QUICK RATIO
402	CUR	CURRENT RATIO
403	SCSH	SALES/CASH
404	SENS	SALES; GENERAL AND ADMINISTRATION/SALES
405	RETU	RECEIVABLES:TURNOVER
406	RIDS	RECEIVABLES:DAYS SALES
407	INTU	INVENTORIES:TURNOVER
408	IIDS	INVENTORIES:DAYS SALES
409	NSWC	NET SALES/WORKING CAPITAL
410	NSNP	NET SALES/PLANT AND EQUIPMENT
411	NSCA	NET SALES/CURRENT ASSETS
412	AST	NET SALES/TOTAL ASSETS
413	NSEMP	NET SALES/EMPLOYEES
414	LTA	TOTAL LIABILITY/TOTAL ASSETS
415	LIC	TOTAL LIABILITY/INVESTED CAPITAL

```
    416   LCE      TOTAL LIABILITY/COMMON EQUITY
    417   TIE      TIMES INTEREST EARNED
    418   CDEQ     CURRENT DEBT/EQUITY
    419   LTDEQ    LONG TERM DEBT/EQUITY
    420   TDEQ     TOTAL DEBT/EQUITY
    421   TAEQ     TOTAL ASSETS/EQUITY
    422   PRNS     PRETAX INCOME/NET SALES
    423   PRTA     PRETAX INCOME/TOTAL ASSETS
    424   PRIC     PRETAX INCOME/INVESTED CAPITAL
    425   PRCE     PRETAX INCOME/COMMON EQUITY
    426   NRNSX    NET INCOME/NET SALES
    427   NRTAX    NET INCOME/TOTAL ASSETS
    428   NRICX    NET INCOME/INVESTED CAPITAL
    429   NRCEX    NET INCOME/COMMON EQUITY
    430   RDNS     RESEARCH AND DEVELOPMENT/NET SALES
    431   RDNI     RESEARCH AND DEVELOPMENT/NET INCOME
    432   RDEMP    RESEARCH AND DEVELOPMENT/EMPLOYEES

STATIC FACTS:
    501   CN       DISCLOSURE COMPANY NUMBER
    502   CO       COMPANY NAME
    503   AD1      STREET ADDRESS 1
    504   AD2      STREET ADDRESS 2
    505   AD3      STREET ADDRESS 3
    506   CY       CITY
    507   ST       STATE
    508   ZP       ZIP CODE
    509   TE       TELEPHONE (NNN-NNN-NNNN)
    510   IN       INCORPORATION
    511   EX       EXCHANGE
    512   TS       TICKER SYMBOL
    513   DUN      DUNS NUMBER (NN-NNN-NNNN)
```

```
514  SC       SIC CODES (CHARACTER)
515  CS       CURRENT OUTSTANDING SHARES + SOURCE
516  AD       AUDITOR
517  ARS      AUDITOR'S REPORT (SHORT)
518  FY       FISCAL YEAR (MMDD)
519  FO       FORTUNE NUMBER
520  CUSIP    CUSIP NUMBER
521  CUSIPI   CUSIP NUMBER (ISSUER PORTION)
522  PC       PRIMARY SIC CODE (NUMERIC)
523  COS      CURRENT OUTSTANDING SHARES
524  SO       SHARES HELD BY OFF + DIR
525  SS       SHAREHOLDERS
526  EM       EMPLOYEES
527  STARTA   ANNUAL START DATE (YYYY)
528  ENDA     ANNUAL END DATE (YYYY)
529  STARTQ   QUARTERLY START DATE (YYYYQQ)
530  ENDQ     QUARTERLY END DATE (YYYYQQ)
531  COMPNO   COMPANY NUMBER
532  USSNO    USSTOCK DATABASE NUMBER (FACT 18)
533  END      DATE LAST DISCLOSURE UPDATE (YYYYMMDD)
534  LAFD     LATEST ANNUAL FINANCIAL DATE
535  LQFD     LATEST QUARTERLY FINANCIAL DATE
536  FN       FORBES NUMBER
537  LC       LEGAL COUNSEL
538  STA      STOCK TRANSFER AGENT
539  NS5      NET SALES (5 YR SUMMARY)
540  NI5      NET INCOME (5 YR SUMMARY)
541  EPS5     EARNINGS PER SHARE (5 YR SUMMARY)
542  GRNS5    5 YEAR GROWTH RATE FOR NET SALES
543  GRNI5    5 YEAR GROWTH RATE FOR NET INCOME
544  GREPS5   5 YEAR GROWTH RATE FOR EPS
601  XR       CROSS REFERENCE
```

```
602  DE       DESCRIPTION OF BUSINESS
603  FL       FILINGS TABLE
604  AC       AUDITOR CHANGE
605  AR       AUDITOR'S REPORT
606  SL       SEGMENT DATA
607  CT       COMMENTS
608  NA       OFFICERS
609  DO       DIRECTORS
610  SH       OWNERSHIP
611  SB       SUBSIDIARIES
612  ES       EXHIBITS
613  OT       OTHER CORPORATE EVENTS
614  TX       MANAGEMENT DISCUSSION
615  PL       PRESIDENTS LETTER
701  OSM      OUTSTANDING SHARES (SOURCE: MULLER)
702  LTRD     LAST TRADE DATE
703  DWE      DATE WEEK END
704  VOL      VOLUME
705  HI       HIGH
706  LOW      LOW
707  CLS      CLOSE
708  PER      PRICE EARNINGS RATIO
709  ENDD     DATE 12 MONTH ENDING
710  EPS      EARNINGS PER SHARE
711  IADIV    INDICATED ANNUAL DIVIDEND
712  FDIV     FIRST DIVIDEND
713  FEXDIV   FIRST EX-DIVIDEND DATE
714  FRD      FIRST RECORD DATE
715  FPD      FIRST PAYABLE DATE
716  FREQ     FREQUENCY
717  PM       PAYMENT METHOD
718  SDIV     SECOND DIVIDEND
```

```
   719   SEXDIV   SECOND EX-DIVIDEND DATE

   720   SRD      SECOND RECORD DATE

   721   SPD      SECOND PAYABLE DATE

   722   SPM      SECOND PAYMENT METHOD

     )OFF

  9898   1989-05-15 17:15:45 IPS

CONNECTED      00:01:37   TO DATE      03:50:59

CPU UNITS        22.656   TO DATE     10106.906

KILOCHARS         7.736   TO DATE       344.821



Lost Carrier


Disconnected

IP Sharp LQP: Started reading result file
IP Sharp LQP - Reading ipresults
     <2 (SYSTEM 0)
      2> (SYSTEM "/usr/cistk/demo/v2/lqp/ipsharp/filter
/usr/cistk/demo/v2/lqp/ipsharp/connect1.tmp")
     <2 (SYSTEM 16777215)
      2> (SYSTEM "/usr/cistk/demo/v2/lqp/ipsharp/preREAD
/usr/cistk/demo/v2/lqp/ipsharp/connect1.tmp")
     <2 (SYSTEM 16777215)
   <1 (SEND-MESSAGE
          (("NO. " "MNEMONIC" "  DESCRIPTION")
           ("  1 " "CH       " "CASH")
           ("  2 " "MS       " "MARKETABLE SECURITIES")
           ("  3 " "RE       " "RECEIVABLES")
           ("  4 " "IV       " "INVENTORIES")
           ("  5 " "RM       " "RAW MATERIALS")
           ("  6 " "WP       " "WORK IN PROGRESS")
           ("  7 " "FG       " "FINISHED GOODS")
           ("  8 " "NR       " "NOTES RECEIVABLE")
           ("  9 " "OC       " "OTHER CURRENT ASSETS")
           (" 10 " "CA       " "CURRENT ASSETS")
           (" 11 " "PR       " "PROPERTY; PLANT + EQUIPMENT")
           (" 12 " "DP       " "ACCUMULATED DEPRECIATION")
           (" 13 " "PN       " "NET PROPERTY; PLANT + EQUIPMENT")
           (" 14 " "IA       " "INVESTMENT + ADVANCES TO SUBS")
           (" 15 " "NC       " "OTHER NON-CURRENT ASSETS")
           (" 16 " "DC       " "DEFERRED CHARGES")
           (" 17 " "IS       " "INTANGIBLES")
           (" 18 " "DS       " "DEPOSITS + OTHER ASSETS")
           (" 19 " "TA       " "TOTAL ASSETS")
           ("101 " "NP       " "NOTES PAYABLE")
           ("102 " "AP       " "ACCOUNTS PAYABLE")
           ("103 " "CD       " "CURRENT LONG TERM DEBT")
           ("104 " "CL       " "CURRENT PORTION OF CAP LEASES")
```

```
("105 " "AE       " "ACCRUED EXPENSES")
("106 " "IC       " "INCOME TAXES")
("107 " "RL       " "OTHER CURRENT LIABILITIES")
("108 " "LI       " "TOTAL CURRENT LIABILITIES")
("109 " "MG       " "MORTGAGES")
("110 " "DF       " "DEFERRED CHARGES/INCOME")
("111 " "CV       " "CONVERTIBLE DEBT")
("112 " "LD       " "LONG TERM DEBT")
("113 " "NL       " "NON-CURRENT CAPITAL LEASES")
("114 " "LL       " "OTHER LONG TERM LIABILITIES")
("115 " "TL       " "TOTAL LIABILITIES")
("116 " "ML       " "MINORITY INTEREST (LIABILITIES)")
("117 " "PS       " "PREFERRED STOCK")
("118 " "SN       " "COMMON STOCK NET")
("119 " "SR       " "CAPITAL SURPLUS")
("120 " "RT       " "RETAINED EARNINGS")
("121 " "TR       " "TREASURY STOCK")
("122 " "OL       " "OTHER LIABILITIES")
("123 " "SE       " "SHAREHOLDER'S EQUITY")
("124 " "LW       " "TOTAL LIABILITIES + NET WORTH")
("201 " "NS       " "NET SALES (INCOME STATEMENT)")
("202 " "CG       " "COST OF GOODS")
("203 " "GP       " "GROSS PROFIT")
("204 " "RD       " "R + D EXPENDITURES")
("205 " "SG       " "SELL; GENERAL + ADMIN EXPENSES")
("206 " "ID       " "INCOME BEFORE DEPREC + AMORT")
("207 " "DA       " "DEPRECIATION + AMORTIZATION")
("208 " "NO       " "NON-OPERATING INCOME")
("209 " "IE       " "INTEREST EXPENSE")
("210 " "IB       " "INCOME BEFORE TAX")
("211 " "PT       " "PROVISION FOR INCOME TAXES")
("212 " "MI       " "MINORITY INTEREST")
("213 " "IG       " "INVESTMENT GAINS/LOSSES")
("214 " "OI       " "OTHER INCOME")
("215 " "IX       " "NET INCOME BEFORE EXTRA ITEMS")
("216 " "EI       " "EXTRA ITEMS + DISCONTINUED OPS")
("217 " "NI       " "NET INCOME (INCOME STATEMENT)")
("218 " "OS       " "OUTSTANDING SHARES")
("301 " "CF       " "FINANCIAL REPORTING DATE (YYYYMMDD)")
("302 " "IBEX     "
  "INCOME (LOSS) BEFORE EXTRAORDINARY ITEMS")
("303 " "DEP      " "DEPRECIATION AND DEPLETION")
("304 " "DEFT     " "DEFERRED INCOME TAX")
("305 " "MININ    "
  "MINORITY INTEREST/EQUITY IN SUBSIDIARIES AND AFFI")
("306 " "OFO      " "OTHER FUNDS FROM OPERATIONS")
("307 " "TFO      " "TOTAL FUNDS PROVIDED BY OPERATIONS")
("308 " "FUEX     " "FUNDS USED FOR EXTRAORDINARY ITEMS")
("309 " "SPPE     " "SALES OF PROPERTY; PLANT AND EQUIPMENT")
("310 " "ILTD     " "ISSUANCE OF LONG TERM DEBT")
("311 " "SOS      " "SALE OF STOCK")
("312 " "OSF      " "OTHER SOURCES OF FUNDS")
("313 " "TSF      " "TOTAL SOURCES OF FUNDS")
("314 " "DIV      " "DIVIDENDS")
("315 " "CEXP     " "CAPITAL EXPENDITURES")
("316 " "ININV    " "INCREASE IN INVESTMENT")
("317 " "DECLTD   " "DECREASE IN LONG TERM DEBT")
("318 " "PURCS    " "PURCHASE OF STOCK")
("319 " "ACQNS    " "ACQUISITIONS")
("320 " "OUF      " "OTHER USES OF FUNDS")
("321 " "TUF      " "TOTAL USES OF FUNDS")
("322 " "INCWC    " "INCREASE (DECREASE) IN WORKING CAPITAL")
("401 " "QKR      " "QUICK RATIO")
("402 " "CUR      " "CURRENT RATIO")
```

```
("403 " "SCSH     " "SALES/CASH")
 ("404 " "SENS     " "SALES; GENERAL AND ADMINISTRATION/SALES")
 ("405 " "RETU     " "RECEIVABLES:TURNOVER")
 ("406 " "RIDS     " "RECEIVABLES:DAYS SALES")
 ("407 " "INTU     " "INVENTORIES:TURNOVER")
 ("408 " "IIDS     " "INVENTORIES:DAYS SALES")
 ("409 " "NSWC     " "NET SALES/WORKING CAPITAL")
 ("410 " "NSNP     " "NET SALES/PLANT AND EQUIPMENT")
 ("411 " "NSCA     " "NET SALES/CURRENT ASSETS")
 ("412 " "AST      " "NET SALES/TOTAL ASSETS")
 ("413 " "NSEMP    " "NET SALES/EMPLOYEES")
 ("414 " "LTA      " "TOTAL LIABILITY/TOTAL ASSETS")
 ("415 " "LIC      " "TOTAL LIABILITY/INVESTED CAPITAL")
 ("416 " "LCE      " "TOTAL LIABILITY/COMMON EQUITY")
("417 " "TIE      " "TIMES INTEREST EARNED")
 ("418 " "CDEQ     " "CURRENT DEBT/EQUITY")
 ("419 " "LTDEQ    " "LONG TERM DEBT/EQUITY")
 ("420 " "TDEQ     " "TOTAL DEBT/EQUITY")
 ("421 " "TAEQ     " "TOTAL ASSETS/EQUITY")
 ("422 " "PRNS     " "PRETAX INCOME/NET SALES")
 ("423 " "PRTA     " "PRETAX INCOME/TOTAL ASSETS")
 ("424 " "PRIC     " "PRETAX INCOME/INVESTED CAPITAL")
 ("425 " "PRCE     " "PRETAX INCOME/COMMON EQUITY")
 ("426 " "NRNSX    " "NET INCOME/NET SALES")
 ("427 " "NRTAX    " "NET INCOME/TOTAL ASSETS")
 ("428 " "NRICX    " "NET INCOME/INVESTED CAPITAL")
 ("429 " "NRCEX    " "NET INCOME/COMMON EQUITY")
 ("430 " "RDNS     " "RESEARCH AND DEVELOPMENT/NET SALES")
 ("431 " "RDNI     " "RESEARCH AND DEVELOPMENT/NET INCOME")
 ("432 " "RDEMP    " "RESEARCH AND DEVELOPMENT/EMPLOYEES")
("501 " "CN       " "DISCLOSURE COMPANY NUMBER")
("502 " "CO       " "COMPANY NAME")
("503 " "AD1      " "STREET ADDRESS 1")
("504 " "AD2      " "STREET ADDRESS 2")
("505 " "AD3      " "STREET ADDRESS 3")
("506 " "CY       " "CITY") ("507 " "ST       " "STATE")
("508 " "ZP       " "ZIP CODE")
 ("509 " "TE       " "TELEPHONE (NNN-NNN-NNNN)")
("510 " "IN       " "INCORPORATION")
("511 " "EX       " "EXCHANGE")
("512 " "TS       " "TICKER SYMBOL")
 ("513 " "DUN      " "DUNS NUMBER (NN-NNN-NNNN)")
("514 " "SC       " "SIC CODES (CHARACTER)")
 ("515 " "CS       " "CURRENT OUTSTANDING SHARES + SOURCE")
("516 " "AD       " "AUDITOR")
 ("517 " "ARS      " "AUDITOR'S REPORT (SHORT)")
("518 " "FY       " "FISCAL YEAR (MMDD)")
("519 " "FO       " "FORTUNE NUMBER")
 ("520 " "CUSIP    " "CUSIP NUMBER")
 ("521 " "CUSIPI   " "CUSIP NUMBER (ISSUER PORTION)")
 ("522 " "PC       " "PRIMARY SIC CODE (NUMERIC)")
 ("523 " "COS      " "CURRENT OUTSTANDING SHARES")
("524 " "SO       " "SHARES HELD BY OFF + DIR")
("525 " "SS       " "SHAREHOLDERS")
("526 " "EM       " "EMPLOYEES")
 ("527 " "STARTA   " "ANNUAL START DATE (YYYY)")
 ("528 " "ENDA     " "ANNUAL END DATE (YYYY)")
 ("529 " "STARTQ   " "QUARTERLY START DATE (YYYYQQ)")
 ("530 " "ENDQ     " "QUARTERLY END DATE (YYYYQQ)")
 ("531 " "COMPNO   " "COMPANY NUMBER")
 ("532 " "USSNO    " "USSTOCK DATABASE NUMBER (FACT 18)")
 ("533 " "END      " "DATE LAST DISCLOSURE UPDATE (YYYYMMDD)")
 ("534 " "LAFD     " "LATEST ANNUAL FINANCIAL DATE")
 ("535 " "LQFD     " "LATEST QUARTERLY FINANCIAL DATE")
```

```
            ("536 " "FN       " "FORBES NUMBER")
            ("537 " "LC       " "LEGAL COUNSEL")
            ("538 " "STA      " "STOCK TRANSFER AGENT")
            ("539 " "NS5      " "NET SALES (5 YR SUMMARY)")
            ("540 " "NI5      " "NET INCOME (5 YR SUMMARY)")
            ("541 " "EPS5     " "EARNINGS PER SHARE (5 YR SUMMARY)")
            ("542 " "GRNS5    " "5 YEAR GROWTH RATE FOR NET SALES")
            ("543 " "GRNI5    " "5 YEAR GROWTH RATE FOR NET INCOME")
            ("544 " "GREPS5   " "5 YEAR GROWTH RATE FOR EPS")
            ("601 " "XR       " "CROSS REFERENCE")
            ("602 " "DE       " "DESCRIPTION OF BUSINESS")
            ("603 " "FL       " "FILINGS TABLE")
            ("604 " "AC       " "AUDITOR CHANGE")
            ("605 " "AR       " "AUDITOR'S REPORT")
            ("606 " "SL       " "SEGMENT DATA")
            ("607 " "CT       " "COMMENTS") ("608 " "NA       " "OFFICERS")
            ("609 " "DO       " "DIRECTORS")
            ("610 " "SH       " "OWNERSHIP")
            ("611 " "SB       " "SUBSIDIARIES")
            ("612 " "ES       " "EXHIBITS")
            ("613 " "OT       " "OTHER CORPORATE EVENTS")
            ("614 " "TX       " "MANAGEMENT DISCUSSION")
            ("615 " "PL       " "PRESIDENTS LETTER")
            ("701 " "OSM      " "OUTSTANDING SHARES (SOURCE: MULLER)")
            ("702 " "LTRD     " "LAST TRADE DATE")
            ("703 " "DWE      " "DATE WEEK END")
            ("704 " "VOL      " "VOLUME") ("705 " "HI       " "HIGH")
            ("706 " "LOW      " "LOW") ("707 " "CLS      " "CLOSE")
            ("708 " "PER      " "PRICE EARNINGS RATIO")
            ("709 " "ENDD     " "DATE 12 MONTH ENDING")
            ("710 " "EPS      " "EARNINGS PER SHARE")
            ("711 " "IADIV    " "INDICATED ANNUAL DIVIDEND")
            ("712 " "FDIV     " "FIRST DIVIDEND")
            ("713 " "FEXDIV   " "FIRST EX-DIVIDEND DATE")
            ("714 " "FRD      " "FIRST RECORD DATE")
            ("715 " "FPD      " "FIRST PAYABLE DATE")
            ("716 " "FREQ     " "FREQUENCY")
            ("717 " "PM       " "PAYMENT METHOD")
            ("718 " "SDIV     " "SECOND DIVIDEND")
            ("719 " "SEXDIV   " "SECOND EX-DIVIDEND DATE")
            ("720 " "SRD      " "SECOND RECORD DATE")
            ("721 " "SPD      " "SECOND PAYABLE DATE")
            ("722 " "SPM      " "SECOND PAYMENT METHOD")))
(("NO. " "MNEMONIC" "  DESCRIPTION") ("  1 " "CH       " "CASH")
 ("  2 " "MS       " "MARKETABLE SECURITIES")
 ("  3 " "RE       " "RECEIVABLES") ("  4 " "IV       " "INVENTORIES")
 ("  5 " "RM       " "RAW MATERIALS")
 ("  6 " "WP       " "WORK IN PROGRESS")
 ("  7 " "FG       " "FINISHED GOODS")
 ("  8 " "NR       " "NOTES RECEIVABLE")
 ("  9 " "OC       " "OTHER CURRENT ASSETS")
 (" 10 " "CA       " "CURRENT ASSETS")
 (" 11 " "PR       " "PROPERTY; PLANT + EQUIPMENT")
 (" 12 " "DP       " "ACCUMULATED DEPRECIATION")
 (" 13 " "PN       " "NET PROPERTY; PLANT + EQUIPMENT")
 (" 14 " "IA       " "INVESTMENT + ADVANCES TO SUBS")
 (" 15 " "NC       " "OTHER NON-CURRENT ASSETS")
 (" 16 " "DC       " "DEFERRED CHARGES")
 (" 17 " "IS       " "INTANGIBLES")
 (" 18 " "DS       " "DEPOSITS + OTHER ASSETS")
 (" 19 " "TA       " "TOTAL ASSETS") ("101 " "NP       " "NOTES PAYABLE")
 ("102 " "AP       " "ACCOUNTS PAYABLE")
 ("103 " "CD       " "CURRENT LONG TERM DEBT")
 ("104 " "CL       " "CURRENT PORTION OF CAP LEASES")
```

```
("105 " "AE        " "ACCRUED EXPENSES")
("106 " "IC        " "INCOME TAXES")
("107 " "RL        " "OTHER CURRENT LIABILITIES")
("108 " "LI        " "TOTAL CURRENT LIABILITIES")
("109 " "MG        " "MORTGAGES")
("110 " "DF        " "DEFERRED CHARGES/INCOME")
("111 " "CV        " "CONVERTIBLE DEBT")
("112 " "LD        " "LONG TERM DEBT")
("113 " "NL        " "NON-CURRENT CAPITAL LEASES")
("114 " "LL        " "OTHER LONG TERM LIABILITIES")
("115 " "TL        " "TOTAL LIABILITIES")
("116 " "ML        " "MINORITY INTEREST (LIABILITIES)")
("117 " "PS        " "PREFERRED STOCK")
("118 " "SN        " "COMMON STOCK NET")
("119 " "SR        " "CAPITAL SURPLUS")
("120 " "RT        " "RETAINED EARNINGS")
("121 " "TR        " "TREASURY STOCK")
("122 " "OL        " "OTHER LIABILITIES")
("123 " "SE        " "SHAREHOLDER'S EQUITY")
("124 " "LW        " "TOTAL LIABILITIES + NET WORTH")
("201 " "NS        " "NET SALES (INCOME STATEMENT)")
("202 " "CG        " "COST OF GOODS") ("203 " "GP        " "GROSS PROFIT")
("204 " "RD        " "R + D EXPENDITURES")
("205 " "SG        " "SELL; GENERAL + ADMIN EXPENSES")
("206 " "ID        " "INCOME BEFORE DEPREC + AMORT")
("207 " "DA        " "DEPRECIATION + AMORTIZATION")
("208 " "NO        " "NON-OPERATING INCOME")
("209 " "IE        " "INTEREST EXPENSE")
("210 " "IB        " "INCOME BEFORE TAX")
("211 " "PT        " "PROVISION FOR INCOME TAXES")
("212 " "MI        " "MINORITY INTEREST")
("213 " "IG        " "INVESTMENT GAINS/LOSSES")
("214 " "OI        " "OTHER INCOME")
("215 " "IX        " "NET INCOME BEFORE EXTRA ITEMS")
("216 " "EI        " "EXTRA ITEMS + DISCONTINUED OPS")
("217 " "NI        " "NET INCOME (INCOME STATEMENT)")
("218 " "OS        " "OUTSTANDING SHARES")
("301 " "CF        " "FINANCIAL REPORTING DATE (YYYYMMDD)")
("302 " "IBEX      " "INCOME (LOSS) BEFORE EXTRAORDINARY ITEMS")
("303 " "DEP       " "DEPRECIATION AND DEPLETION")
("304 " "DEFT      " "DEFERRED INCOME TAX")
("305 " "MININ    "
 "MINORITY INTEREST/EQUITY IN SUBSIDIARIES AND AFFI")
("306 " "OFO       " "OTHER FUNDS FROM OPERATIONS")
("307 " "TFO       " "TOTAL FUNDS PROVIDED BY OPERATIONS")
("308 " "FUEX      " "FUNDS USED FOR EXTRAORDINARY ITEMS")
("309 " "SPPE      " "SALES OF PROPERTY; PLANT AND EQUIPMENT")
("310 " "ILTD      " "ISSUANCE OF LONG TERM DEBT")
("311 " "SOS       " "SALE OF STOCK")
("312 " "OSF       " "OTHER SOURCES OF FUNDS")
("313 " "TSF       " "TOTAL SOURCES OF FUNDS")
("314 " "DIV       " "DIVIDENDS")
("315 " "CEXP      " "CAPITAL EXPENDITURES")
("316 " "ININV     " "INCREASE IN INVESTMENT")
("317 " "DECLTD    " "DECREASE IN LONG TERM DEBT")
("318 " "PURCS     " "PURCHASE OF STOCK")
("319 " "ACQNS     " "ACQUISITIONS")
("320 " "OUF       " "OTHER USES OF FUNDS")
("321 " "TUF       " "TOTAL USES OF FUNDS")
("322 " "INCWC     " "INCREASE (DECREASE) IN WORKING CAPITAL")
("401 " "QKR       " "QUICK RATIO") ("402 " "CUR       " "CURRENT RATIO")
("403 " "SCSH      " "SALES/CASH")
("404 " "SENS      " "SALES; GENERAL AND ADMINISTRATION/SALES")
("405 " "RETU      " "RECEIVABLES:TURNOVER")
```

```
("406 " "RIDS     " "RECEIVABLES:DAYS SALES")
("407 " "INTU     " "INVENTORIES:TURNOVER")
("408 " "IIDS     " "INVENTORIES:DAYS SALES")
("409 " "NSWC     " "NET SALES/WORKING CAPITAL")
("410 " "NSNP     " "NET SALES/PLANT AND EQUIPMENT")
("411 " "NSCA     " "NET SALES/CURRENT ASSETS")
("412 " "AST      " "NET SALES/TOTAL ASSETS")
("413 " "NSEMP    " "NET SALES/EMPLOYEES")
("414 " "LTA      " "TOTAL LIABILITY/TOTAL ASSETS")
("415 " "LIC      " "TOTAL LIABILITY/INVESTED CAPITAL")
("416 " "LCE      " "TOTAL LIABILITY/COMMON EQUITY")
("417 " "TIE      " "TIMES INTEREST EARNED")
("418 " "CDEQ     " "CURRENT DEBT/EQUITY")
("419 " "LTDEQ    " "LONG TERM DEBT/EQUITY")
("420 " "TDEQ     " "TOTAL DEBT/EQUITY")
("421 " "TAEQ     " "TOTAL ASSETS/EQUITY")
("422 " "PRNS     " "PRETAX INCOME/NET SALES")
("423 " "PRTA     " "PRETAX INCOME/TOTAL ASSETS")
("424 " "PRIC     " "PRETAX INCOME/INVESTED CAPITAL")
("425 " "PRCE     " "PRETAX INCOME/COMMON EQUITY")
("426 " "NRNSX    " "NET INCOME/NET SALES")
("427 " "NRTAX    " "NET INCOME/TOTAL ASSETS")
("428 " "NRICX    " "NET INCOME/INVESTED CAPITAL")
("429 " "NRCEX    " "NET INCOME/COMMON EQUITY")
("430 " "RDNS     " "RESEARCH AND DEVELOPMENT/NET SALES")
("431 " "RDNI     " "RESEARCH AND DEVELOPMENT/NET INCOME")
("432 " "RDEMP    " "RESEARCH AND DEVELOPMENT/EMPLOYEES")
("501 " "CN       " "DISCLOSURE COMPANY NUMBER")
("502 " "CO       " "COMPANY NAME")
("503 " "AD1      " "STREET ADDRESS 1")
("504 " "AD2      " "STREET ADDRESS 2")
("505 " "AD3      " "STREET ADDRESS 3") ("506 " "CY       " "CITY")
("507 " "ST       " "STATE") ("508 " "ZP       " "ZIP CODE")
("509 " "TE       " "TELEPHONE (NNN-NNN-NNNN)")
("510 " "IN       " "INCORPORATION") ("511 " "EX       " "EXCHANGE")
("512 " "TS       " "TICKER SYMBOL")
("513 " "DUN      " "DUNS NUMBER (NN-NNN-NNNN)")
("514 " "SC       " "SIC CODES (CHARACTER)")
("515 " "CS       " "CURRENT OUTSTANDING SHARES + SOURCE")
("516 " "AD       " "AUDITOR")
("517 " "ARS      " "AUDITOR'S REPORT (SHORT)")
("518 " "FY       " "FISCAL YEAR (MMDD)")
("519 " "FO       " "FORTUNE NUMBER")
("520 " "CUSIP    " "CUSIP NUMBER")
("521 " "CUSIPI   " "CUSIP NUMBER (ISSUER PORTION)")
("522 " "PC       " "PRIMARY SIC CODE (NUMERIC)")
("523 " "COS      " "CURRENT OUTSTANDING SHARES")
("524 " "SO       " "SHARES HELD BY OFF + DIR")
("525 " "SS       " "SHAREHOLDERS") ("526 " "EM       " "EMPLOYEES")
("527 " "STARTA   " "ANNUAL START DATE (YYYY)")
("528 " "ENDA     " "ANNUAL END DATE (YYYY)")
("529 " "STARTQ   " "QUARTERLY START DATE (YYYYQQ)")
("530 " "ENDQ     " "QUARTERLY END DATE (YYYYQQ)")
("531 " "COMPNO   " "COMPANY NUMBER")
("532 " "USSNO    " "USSTOCK DATABASE NUMBER (FACT 18)")
("533 " "END      " "DATE LAST DISCLOSURE UPDATE (YYYYMMDD)")
("534 " "LAFD     " "LATEST ANNUAL FINANCIAL DATE")
("535 " "LQFD     " "LATEST QUARTERLY FINANCIAL DATE")
("536 " "FN       " "FORBES NUMBER")
("537 " "LC       " "LEGAL COUNSEL")
("538 " "STA      " "STOCK TRANSFER AGENT")
("539 " "NS5      " "NET SALES (5 YR SUMMARY)")
("540 " "NI5      " "NET INCOME (5 YR SUMMARY)")
("541 " "EPS5     " "EARNINGS PER SHARE (5 YR SUMMARY)")
```

```
 ("542 " "GRNS5    " "5 YEAR GROWTH RATE FOR NET SALES")
 ("543 " "GRNI5    " "5 YEAR GROWTH RATE FOR NET INCOME")
 ("544 " "GREPS5   " "5 YEAR GROWTH RATE FOR EPS")
 ("601 " "XR       " "CROSS REFERENCE")
 ("602 " "DE       " "DESCRIPTION OF BUSINESS")
 ("603 " "FL       " "FILINGS TABLE")
 ("604 " "AC       " "AUDITOR CHANGE")
 ("605 " "AR       " "AUDITOR'S REPORT")
 ("606 " "SL       " "SEGMENT DATA") ("607 " "CT       " "COMMENTS")
 ("608 " "NA       " "OFFICERS") ("609 " "DO       " "DIRECTORS")
 ("610 " "SH       " "OWNERSHIP") ("611 " "SB       " "SUBSIDIARIES")
 ("612 " "ES       " "EXHIBITS")
 ("613 " "OT       " "OTHER CORPORATE EVENTS")
 ("614 " "TX       " "MANAGEMENT DISCUSSION")
 ("615 " "PL       " "PRESIDENTS LETTER")
 ("701 " "OSM      " "OUTSTANDING SHARES (SOURCE: MULLER)")
 ("702 " "LTRD     " "LAST TRADE DATE")
 ("703 " "DWE      " "DATE WEEK END") ("704 " "VOL      " "VOLUME")
 ("705 " "HI       " "HIGH") ("706 " "LOW      " "LOW")
 ("707 " "CLS      " "CLOSE") ("708 " "PER      " "PRICE EARNINGS RATIO")
 ("709 " "ENDD     " "DATE 12 MONTH ENDING")
 ("710 " "EPS      " "EARNINGS PER SHARE")
 ("711 " "IADIV    " "INDICATED ANNUAL DIVIDEND")
 ("712 " "FDIV     " "FIRST DIVIDEND")
 ("713 " "FEXDIV   " "FIRST EX-DIVIDEND DATE")
 ("714 " "FRD      " "FIRST RECORD DATE")
 ("715 " "FPD      " "FIRST PAYABLE DATE")
 ("716 " "FREQ     " "FREQUENCY") ("717 " "PM       " "PAYMENT METHOD")
 ("718 " "SDIV     " "SECOND DIVIDEND")
 ("719 " "SEXDIV   " "SECOND EX-DIVIDEND DATE")
 ("720 " "SRD      " "SECOND RECORD DATE")
 ("721 " "SPD      " "SECOND PAYABLE DATE")
 ("722 " "SPM      " "SECOND PAYMENT METHOD"))

>(printf   lqp-ipsharp4)

(SEND-MESSAGE 'DISCLOSUREDB :GET-TABLES)
(SEND-MESSAGE 'DISCLOSUREDB :GET-TABLES)

>(eval lqp-ipsharp4)
   1> (SEND-MESSAGE DISCLOSUREDB :GET-TABLES)
      2> (SYSTEM "\"/usr/cistk/demo/v2/lqp/ipsharp/qt\" \"foreign\" \"SLOAN:...\"
\"...\" h19 \"/usr/bin\" \"isql adisclosure -\" \"INFO TABLES\"")
IPSharp LQP - Query Received: INFO TABLES
Finished reading file
Connected



)


  GHBGSBHSGBSBGBSGBM4M4M4M4MNMNMNMNM58585858HEHEHEHEH)1769739:SLOAN


   9986) 1989-05-15 17:20:41 IPSA


SHARP APL SERVICE


       )LOAD 39 MAGIC
```

```
SAVED   1989-05-15 01:44:37

       INFOMAGIC


WELCOME TO INFOMAGIC.  ENTER  DESCRIBE  FOR DOCUMENTATION.


SELECT SERVICE:
 LIST SERVICES

 - LIST SERVICES -


INFOMAGIC SERVICES  -  LISTING OF ALL SERVICES AVAILABLE



THE FOLLOWING SERVICES ARE AVAILABLE IN THE CATEGORY OF FINANCE


AUSSTOCK           - STOCK DATA FOR ISSUES TRADING ON THE AUSTRALIAN STOCK EXCHANG
CANADIAN BANKS     - CANADIAN BANKS FILING SCHEDULES J. AND O.
CANOILS            - CANADIAN OILS CORPORATE DATA, FINANCIAL AND OPERATING
CURRENCY           - DAILY CURRENCY EXCHANGE RATES
DISCLOSURE         - U.S. CORPORATE STATISTICS FOR 9500 PUBLIC COMPANIES
FPCORP             - CANADIAN CORPORATE STATISTICS FROM THE FINANCIAL POST
FXPRO              - FOREIGN CURRENCY PROJECTIONS FROM S.J. RUNDT
HKSTOCK            - STOCK DATA FOR ISSUES TRADING ON THE HONG KONG STOCK EXCHANGE
MARKETSCOPE        - STANDARD AND POORS MARKETSCOPE
NASTOCK            - STOCK DATA FOR ISSUES TRADING ON CANADIAN STOCK EXCHANGES
PRICELINK          - FINANCIAL PRICE DELIVERY SYSTEM
SINGSTOCK          - STOCK DATA FOR ISSUES TRADING ON THE SINGAPORE STOCK EXCHANGE
STATEX             - CORPORATE STATISTICS FOR SYDNEY STOCK EXCHANGE COMPANIES
STOCKMARKETS       - SECURITIES DATA FOR 7 INTERNATIONAL STOCK MARKETS
SYDSTOCK           - STOCK DATA FOR ISSUES TRADING ON THE SYDNEY STOCK EXCHANGE
TSE300             - STOCK DATA FOR ISSUES INCLUDED IN THE TSE 300
```

USSTOCK - STOCK DATA FOR ALL U.S. MAJOR AND REGIONAL EXCHANGES

THE FOLLOWING SERVICES ARE AVAILABLE IN THE CATEGORY OF ECONOMICS

AGWEEKLY - ALBERTA AGRICULTURE NEWSLETTER

CANSIM - CANADIAN ECONOMIC STATISTICS FROM STATISTICS CANADA

CEP - COUNTRY ECONOMIC PROFILES

CITIBASE - U.S. CURRENT ECONOMIC AND FINANCIAL STATISTICS

DOT - DIRECTION OF TRADE STATISTICS

IFS - INTERNATIONAL FINANCIAL STATISTICS

IIF - INTERNATIONAL INSTITUTE OF FINANCE DATA BASE (VERSION 2)

NPA - NATIONAL PLANNING ASSOCIATION DATA BASES

SITC - UNITED NATIONS COMMODITY TRADE STATISTICS

SJRUNDT - COUNTRY RISK ANALYSIS REPORTS FROM S.J. RUNDT

THE FOLLOWING SERVICES ARE AVAILABLE IN THE CATEGORY OF AVIATION

AOCI - AIRPORT OPERATORS COUNCIL INT'L A.C.I.S. SYSTEM

AVDAILY - AVIATION DAILY INDUSTRY PERFORMANCE ANALYSIS SYSTEM

FORM41 - FINANCIAL AND TRAFFIC STATS FOR U.S. CERTIFICATED AIRLINES

OAGEE - OFFICIAL AIRLINE GUIDES - ELECTRONIC EDITION

OAG - OFFICIAL AIRLINE GUIDE DATA BASE

OAND - ORIGIN AND DESTINATION DATABASE (TABLES 8 AND 10)

THE FOLLOWING SERVICES ARE AVAILABLE IN THE CATEGORY OF ENERGY

API - AMERICAN PETROLEUM INSTITUTE STATISTICS

ARGUS - PETROLEUM ARGUS DAILY REPORT

CMAI - CHEMICAL MARKET ASSOCIATES PETROCHEMICAL REPORTS

DEWITT - DEWITT PETROCHEMICAL NEWSLETTERS

HUGHES - EXPLORATORY RIG STATISTICS FROM THE HUGHES TOOL COMPANY

LUNDBERG - U.S. FUEL DATA AS COLLECTED BY LUNDBERG SURVEY INC

```
PETROFLASH   - OIL MARKET AND PRICE INFORMATION SERVICES
PIPELINE     - REUTER WORLD ENERGY PIPELINE
PIW          - PETROLEUM INTELLIGENCE WEEKLY
STATSCAN     - STATISTICS CANADA ENERGY INFORMATION
TECNON       - TECNON (UK) LIMITED PETROCHEMICAL NEWSLETTERS
USDOE        - UNITED STATES DEPT OF ENERGY STATISTICS

THE FOLLOWING SERVICES ARE AVAILABLE IN THE CATEGORY OF NEWS SERVICES

DOW JONES  - DOW JONES NEWS/RETRIEVAL

THE FOLLOWING SERVICES ARE AVAILABLE IN THE CATEGORY OF USER SERVICES

CC                        - CAMPUS CONNECTIONS UNIVERSITY GRADUATE DATABANK
INVOICE                   - PREVIOUS MONTH'S INFOMAGIC INVOICE
KEYWORDS                  - LIST OF SPECIAL COMMANDS AVAILABLE AT INFOMAGIC PROMPTS
LIST SERVICES             - LISTING OF ALL INFOMAGIC SERVICES AVAILABLE
MAIL                      - I.P. SHARP ELECTRONIC MAIL SYSTEM
MANUALS                   - I.P. SHARP PUBLICATIONS ORDERING SYSTEM
SEARCH                    - KEYWORD SEARCH ON ALL INFOMAGIC REPORT TITLES
SUGGESTIONS               - SUGGESTION BOX FOR USER COMMENTS CONCERNING INFOMAGIC
WEIGHTS AND MEASURES   - WEIGHT AND MEASUREMENT CONVERSIONS

SELECT SERVICE:
 )OFF

      )OFF

   9986  1989-05-15 17:21:44 IPS
CONNECTED      00:01:03  TO DATE      03:52:02
CPU UNITS        82.453  TO DATE   10189.359
KILOCHARS         4.198  TO DATE     349.019
```

```
Lost Carrier


Disconnected

IP Sharp LQP: Started reading result file
IP Sharp LQP - Reading ipresults
     <2 (SYSTEM 0)
      2> (SYSTEM "/usr/cistk/demo/v2/lqp/ipsharp/filter
/usr/cistk/demo/v2/lqp/ipsharp/connect1.tmp")
     <2 (SYSTEM 16777215)
      2> (SYSTEM "/usr/cistk/demo/v2/lqp/ipsharp/preREAD
/usr/cistk/demo/v2/lqp/ipsharp/connect1.tmp")
     <2 (SYSTEM 16777215)
   <1 (SEND-MESSAGE
          (("AUSSTOCK"
             "STOCK DATA FOR ISSUES TRADING ON THE AUSTRALIAN STOCK EXCHANG")
           ("CANADIAN BANKS"
            "CANADIAN BANKS FILING SCHEDULES J. AND O.")
           ("CANOILS"
             "CANADIAN OILS CORPORATE DATA, FINANCIAL AND OPERATING")
            ("CURRENCY" "DAILY CURRENCY EXCHANGE RATES")
           ("DISCLOSURE"
             "U.S. CORPORATE STATISTICS FOR 9500 PUBLIC COMPANIES")
           ("FPCORP"
             "CANADIAN CORPORATE STATISTICS FROM THE FINANCIAL POST")
            ("FXPRO" "FOREIGN CURRENCY PROJECTIONS FROM S.J. RUNDT")
           ("HKSTOCK"
             "STOCK DATA FOR ISSUES TRADING ON THE HONG KONG STOCK EXCHANGE")
            ("MARKETSCOPE" "STANDARD AND POORS MARKETSCOPE")
           ("NASTOCK"
             "STOCK DATA FOR ISSUES TRADING ON CANADIAN STOCK EXCHANGES")
            ("PRICELINK" "FINANCIAL PRICE DELIVERY SYSTEM")
           ("SINGSTOCK"
             "STOCK DATA FOR ISSUES TRADING ON THE SINGAPORE STOCK EXCHANGE")
           ("STATEX"
             "CORPORATE STATISTICS FOR SYDNEY STOCK EXCHANGE COMPANIES")
           ("STOCKMARKETS"
             "SECURITIES DATA FOR 7 INTERNATIONAL STOCK MARKETS")
           ("SYDSTOCK"
             "STOCK DATA FOR ISSUES TRADING ON THE SYDNEY STOCK EXCHANGE")
            ("TSE300" "STOCK DATA FOR ISSUES INCLUDED IN THE TSE 300")
           ("USSTOCK"
             "STOCK DATA FOR ALL U.S. MAJOR AND REGIONAL EXCHANGES")
            ("AGWEEKLY" "ALBERTA AGRICULTURE NEWSLETTER")
           ("CANSIM"
             "CANADIAN ECONOMIC STATISTICS FROM STATISTICS CANADA")
            ("CEP" "COUNTRY ECONOMIC PROFILES")
            ("CITIBASE" "U.S. CURRENT ECONOMIC AND FINANCIAL STATISTICS")
            ("DOT" "DIRECTION OF TRADE STATISTICS")
            ("IFS" "INTERNATIONAL FINANCIAL STATISTICS")
           ("IIF"
             "INTERNATIONAL INSTITUTE OF FINANCE DATA BASE (VERSION 2)")
            ("NPA" "NATIONAL PLANNING ASSOCIATION DATA BASES")
            ("SITC" "UNITED NATIONS COMMODITY TRADE STATISTICS")
            ("SJRUNDT" "COUNTRY RISK ANALYSIS REPORTS FROM S.J. RUNDT")
            ("AOCI" "AIRPORT OPERATORS COUNCIL INT'L  A.C.I.S. SYSTEM")
           ("AVDAILY"
             "AVIATION DAILY INDUSTRY PERFORMANCE ANALYSIS SYSTEM")
           ("FORM41"
             "FINANCIAL AND TRAFFIC STATS FOR U.S. CERTIFICATED AIRLINES")
            ("OAGEE" "OFFICIAL AIRLINE GUIDES - ELECTRONIC EDITION")
```

```
            ("OAG" "OFFICIAL AIRLINE GUIDE DATA BASE")
            ("OAND" "ORIGIN AND DESTINATION DATABASE (TABLES 8 AND 10)")
            ("API" "AMERICAN PETROLEUM INSTITUTE STATISTICS")
            ("ARGUS" "PETROLEUM ARGUS DAILY REPORT")
            ("CMAI" "CHEMICAL MARKET ASSOCIATES PETROCHEMICAL REPORTS")
            ("DEWITT" "DEWITT PETROCHEMICAL NEWSLETTERS")
           ("HUGHES"
             "EXPLORATORY RIG STATISTICS FROM THE HUGHES TOOL COMPANY")
           ("LUNDBERG"
             "U.S. FUEL DATA AS COLLECTED BY LUNDBERG SURVEY INC")
            ("PETROFLASH" "OIL MARKET AND PRICE INFORMATION SERVICES")
            ("PIPELINE" "REUTER WORLD ENERGY PIPELINE")
            ("PIW" "PETROLEUM INTELLIGENCE WEEKLY")
            ("STATSCAN" "STATISTICS CANADA ENERGY INFORMATION")
            ("TECNON" "TECNON (UK) LIMITED PETROCHEMICAL NEWSLETTERS")
            ("USDOE" "UNITED STATES DEPT OF ENERGY STATISTICS")
            ("DOW JONES" "DOW JONES NEWS/RETRIEVAL")
            ("CC" "CAMPUS CONNECTIONS UNIVERSITY GRADUATE DATABANK")
            ("INVOICE" "PREVIOUS MONTH'S INFOMAGIC INVOICE")
           ("KEYWORDS"
             "LIST OF SPECIAL COMMANDS AVAILABLE AT INFOMAGIC PROMPTS")
           ("LIST SERVICES"
             "LISTING OF ALL INFOMAGIC SERVICES AVAILABLE")
            ("MAIL" "I.P. SHARP ELECTRONIC MAIL SYSTEM")
            ("MANUALS" "I.P. SHARP PUBLICATIONS ORDERING SYSTEM")
            ("SEARCH" "KEYWORD SEARCH ON ALL INFOMAGIC REPORT TITLES")
           ("SUGGESTIONS"
             "SUGGESTION BOX FOR USER COMMENTS CONCERNING INFOMAGIC")
            ("WEIGHTS AND MEASURES" "WEIGHT AND MEASUREMENT CONVERSIONS")))
(("AUSSTOCK"
   "STOCK DATA FOR ISSUES TRADING ON THE AUSTRALIAN STOCK EXCHANG")
  ("CANADIAN BANKS" "CANADIAN BANKS FILING SCHEDULES J. AND O.")
  ("CANOILS" "CANADIAN OILS CORPORATE DATA, FINANCIAL AND OPERATING")
  ("CURRENCY" "DAILY CURRENCY EXCHANGE RATES")
  ("DISCLOSURE" "U.S. CORPORATE STATISTICS FOR 9500 PUBLIC COMPANIES")
  ("FPCORP" "CANADIAN CORPORATE STATISTICS FROM THE FINANCIAL POST")
  ("FXPRO" "FOREIGN CURRENCY PROJECTIONS FROM S.J. RUNDT")
  ("HKSTOCK"
   "STOCK DATA FOR ISSUES TRADING ON THE HONG KONG STOCK EXCHANGE")
  ("MARKETSCOPE" "STANDARD AND POORS MARKETSCOPE")
  ("NASTOCK"
   "STOCK DATA FOR ISSUES TRADING ON CANADIAN STOCK EXCHANGES")
  ("PRICELINK" "FINANCIAL PRICE DELIVERY SYSTEM")
  ("SINGSTOCK"
   "STOCK DATA FOR ISSUES TRADING ON THE SINGAPORE STOCK EXCHANGE")
  ("STATEX" "CORPORATE STATISTICS FOR SYDNEY STOCK EXCHANGE COMPANIES")
  ("STOCKMARKETS" "SECURITIES DATA FOR 7 INTERNATIONAL STOCK MARKETS")
  ("SYDSTOCK"
   "STOCK DATA FOR ISSUES TRADING ON THE SYDNEY STOCK EXCHANGE")
  ("TSE300" "STOCK DATA FOR ISSUES INCLUDED IN THE TSE 300")
  ("USSTOCK" "STOCK DATA FOR ALL U.S. MAJOR AND REGIONAL EXCHANGES")
  ("AGWEEKLY" "ALBERTA AGRICULTURE NEWSLETTER")
  ("CANSIM" "CANADIAN ECONOMIC STATISTICS FROM STATISTICS CANADA")
  ("CEP" "COUNTRY ECONOMIC PROFILES")
  ("CITIBASE" "U.S. CURRENT ECONOMIC AND FINANCIAL STATISTICS")
  ("DOT" "DIRECTION OF TRADE STATISTICS")
  ("IFS" "INTERNATIONAL FINANCIAL STATISTICS")
  ("IIF" "INTERNATIONAL INSTITUTE OF FINANCE DATA BASE (VERSION 2)")
  ("NPA" "NATIONAL PLANNING ASSOCIATION DATA BASES")
  ("SITC" "UNITED NATIONS COMMODITY TRADE STATISTICS")
  ("SJRUNDT" "COUNTRY RISK ANALYSIS REPORTS FROM S.J. RUNDT")
  ("AOCI" "AIRPORT OPERATORS COUNCIL INT'L  A.C.I.S. SYSTEM")
  ("AVDAILY" "AVIATION DAILY INDUSTRY PERFORMANCE ANALYSIS SYSTEM")
  ("FORM41"
```

```
 "FINANCIAL AND TRAFFIC STATS FOR U.S. CERTIFICATED AIRLINES")
("OAGEE" "OFFICIAL AIRLINE GUIDES - ELECTRONIC EDITION")
("OAG" "OFFICIAL AIRLINE GUIDE DATA BASE")
("OAND" "ORIGIN AND DESTINATION DATABASE (TABLES 8 AND 10)")
("API" "AMERICAN PETROLEUM INSTITUTE STATISTICS")
("ARGUS" "PETROLEUM ARGUS DAILY REPORT")
("CMAI" "CHEMICAL MARKET ASSOCIATES PETROCHEMICAL REPORTS")
("DEWITT" "DEWITT PETROCHEMICAL NEWSLETTERS")
("HUGHES" "EXPLORATORY RIG STATISTICS FROM THE HUGHES TOOL COMPANY")
("LUNDBERG" "U.S. FUEL DATA AS COLLECTED BY LUNDBERG SURVEY INC")
("PETROFLASH" "OIL MARKET AND PRICE INFORMATION SERVICES")
("PIPELINE" "REUTER WORLD ENERGY PIPELINE")
("PIW" "PETROLEUM INTELLIGENCE WEEKLY")
("STATSCAN" "STATISTICS CANADA ENERGY INFORMATION")
("TECNON" "TECNON (UK) LIMITED PETROCHEMICAL NEWSLETTERS")
("USDOE" "UNITED STATES DEPT OF ENERGY STATISTICS")
("DOW JONES" "DOW JONES NEWS/RETRIEVAL")
("CC" "CAMPUS CONNECTIONS UNIVERSITY GRADUATE DATABANK")
("INVOICE" "PREVIOUS MONTH'S INFOMAGIC INVOICE")
("KEYWORDS" "LIST OF SPECIAL COMMANDS AVAILABLE AT INFOMAGIC PROMPTS")
("LIST SERVICES" "LISTING OF ALL INFOMAGIC SERVICES AVAILABLE")
("MAIL" "I.P. SHARP ELECTRONIC MAIL SYSTEM")
("MANUALS" "I.P. SHARP PUBLICATIONS ORDERING SYSTEM")
("SEARCH" "KEYWORD SEARCH ON ALL INFOMAGIC REPORT TITLES")
("SUGGESTIONS"
 "SUGGESTION BOX FOR USER COMMENTS CONCERNING INFOMAGIC")
("WEIGHTS AND MEASURES" "WEIGHT AND MEASUREMENT CONVERSIONS"))
```